OPTIMIZATION-DRIVEN DATA-BASED CONSTRAINTS IDENTIFICATION VIA EXPLICIT MATHEMATICAL AND IMPLICIT MACHINE-LEARNING-BASED CONSTITUTIVES

by

Abdullah Aladağ

B.S., Chemical Engineering, Yıldız Technical University, 2019

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Chemical Engineering

Boğaziçi University

2023

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere appreciation to my advisor Prof. Uğur Akman, who was beyond an advisor for me. He provided me the opportunity I have experienced; an incredible journey which embraced not only full knowledge of optimization techniques, coding, machine learning, in short, mathematics, but also attainment of ability to assess the world from many different perspectives. His infinite understanding and support for me to be able to gain many academic skills will always stay with me unforgettable wherever I will be in the future.

I am sincerely thankful to committee members of my thesis, Assoc. Prof. Burak Alakent and Assoc. Prof. Devrim Barış Kaymak, for sparing their valuable time for reading and commenting on my thesis.

I am sincerely grateful to Duygu Kaya for her support and valuable friendship since my undergraduate years.

I am extremely grateful to my mentor Serdar Özen, Head of Commercial Planning at Nestlé, for guiding me by sharing his invaluable experiences, which have significantly contributed to my decisions on my career, during my graduate education.

I am also extremely grateful to İbrahim Kadıoğlu, Head of Business Excellence at SOCAR Türkiye, for always supporting young engineers like me and for providing a working environment under the umbrella of business excellence, with the notion of being a well-educated person.

Finally, I would like to present my sincere indebtedness to my father and mother, Bahadır Aladağ and Nermin Aladağ, who have equipped me with a philosophy of how to react to what have happened around me, the viewpoint which has formed the basis of my decisions, and have always stood behind me to pursue a life path I believe. I am also grateful to my sister Zeynep Aladağ for being with me during all stages of my life.

ABSTRACT

OPTIMIZATION-DRIVEN DATA-BASED CONSTRAINTS IDENTIFICATION VIA EXPLICIT MATHEMATICAL AND IMPLICIT MACHINE-LEARNING-BASED CONSTITUTIVES

The major aim of “data-based constraint identification” is to identify feasible regions within which a process can be operated. Our approach is based on the quantitative-feasibility information of sample points metamorphosed into single- and multiple-mathematical equations constituting the data-based constraints. We firstly devise an “overall objective function” which is capable of identifying feasible regions with multiple-constitutive inequality constraints by resorting to the technique of “constraint aggregation”. We then equip our algorithm with the “form-specific constitutives” build via the generic mathematical description of some plausible inequality constraints such as the bound, linear, circular, and ellipsoidal, as single or aggregated multiple constitutives. We then build the “form-specific” and “form-free” constitutives via the “design matrix” approach, also as single or aggregated multiple constitutives. We devise the “implicit neural constitutives” as well via some Machine Learning algorithms such as Neural Networks and Extreme Learning Machines, as single implicit or aggregated multiple implicit constitutives. All of these data-based constitutive constraints are generic such that they can identify N-dimensional feasible regions. We solve the demonstrative examples with the Differential Evolution or Covariance Matrix Adaptation Evolution Strategy global optimizers. Via many diversified examples, including several chemical-engineering related ones, we show that our algorithm can identify joint, disjoint, convex, or nonconvex regions or their combinations. We also apply classification techniques, such as Probabilistic Neural Network, k-Nearest Neighbour, Support Vector Machine, Gaussian Process Regression, and Regression Trees to constraint identification. Our algorithm is also successful in identifying constraints from image boundaries, i.e., in “image-to-constraints” conversion tasks.

ÖZET

AÇIK MATEMATİKSEL VE ÖRTÜLÜ MAKİNE ÖĞRENMESİ TEMELLİ KURUCU ÖZGENLERLE ENİYİLEME ÖNCÜLLÜ VERİ TABANLI KISITLAR ÖZDEŞİMİ

"Veri tabanlı kısıt belirleme" çalışmasının temel amacı, bir sürecin işletilebileceği uygulanabilir bölgeleri saptamaktır. Yaklaşımımız, veri tabanlı kısıtları oluşturan tekli ve çoklu matematiksel denklemlere dönüştürülen örnek noktalarının niceliksel uygunluk bilgisine dayanmaktadır. İlk olarak, "kısıt birleştirme" tekniğine başvurarak çoklu kurucu eşitsizlik kısıtları ile olurlu bölgeleri tanımlayabilen bir "bütünsel amaç fonksiyonu" geliştirdik. Daha sonra, sınır, lineer, dairesel ve elipsoidal gibi bazı olası eşitsizlik kısıtlarının genel matematiksel tanımlarından yola çıkarak tekli ya da birleştirilmiş çoklu kurucular olabilecek şekilde kurgulanan "şekle özgü kurucular" özelliğini algoritmamıza ekledik. Yine tekli yada birleştirilmiş çoklu kurucular olabilen "şekle özgü" ve "şekilden bağımsız" kurucuları "tasarım matrisi" yaklaşımı ile oluşturduk. Ek olarak, Sinir Ağları, Ekstrem Öğrenme Makineleri gibi bazı makine öğrenmesi algoritmaları ile tekli örtülü ya da birleştirilmiş çoklu örtülü kurucular olabilen "örtülü sinir kurucularını" tasarladık. Bu veri tabanlı kurucu kısıtlarının hepsi D-boyutlu uygulanabilir bölgeleri belirleyebilecek şekilde genelleyicidir. Örnekleri Diferansiyel Evrim ve Kovaryans Matris Uyarlama Evrim Stratejisi global eniyileyiciler ile çözdük. Algoritmamızın birleşik, ayrık, konveks ya da konveks olmayan ya da bunların karmalarını saptayabildiğini, kimya mühendisliği ile ilgili örnekleri de içeren birçok çeşitli örnek üzerinden, gösterdik. Ayrıca, Olasılıksal Sinir Ağı, k-En Yakın Komşu, Destek Vektör Makinesi, Gauss Süreç Regresyonu ve Regresyon Ağaçları gibi sınıflandırma tekniklerini kısıt belirlemeye uyguladık. Algoritmamız ayrıca görüntü sınırlarından kısıtların belirlenmesinde, yani "görüntüden kısıtlara" dönüşüm görevinde de başarılı olmuştur.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\boldsymbol{\mathcal{B}}^{\mathcal{H}}$	Bias vector of hidden layer
$\boldsymbol{\mathcal{B}}^{\mathcal{O}}$	Bias vector of output layer
$\mathrm{C}_{\mathrm{KS}}(\mathrm{g}, \rho)$	Functional form of Kreisselmeier-Steinhauser function
D	Dimension of sample points
$\mathbf{D_{XY}}$	Distance matrix
F^{f}	Fraction of feasible points
F^{i}	Fraction of infeasible points
$\mathrm{f}^{*}(\boldsymbol{\beta})$	Updated objective function
g	Inequality constraint function
$\mathbf{g}(\mathbf{X})$	Inequality constraints evaluated at feasible points
$\mathbf{g}(\mathbf{Y})$	Inequality constraints evaluated at infeasible points
$\mathbf{g}(\mathbf{z})$	Inequality constraints evaluated at sample point
$\mathbf{g}\left(\mathbf{z}_1^{\mathrm{b}}, \mathbf{z}_2^{\mathrm{b}}\right)$	Inequality constraints evaluated at two-dimensional feasible points
$\mathbf{g}(\mathbf{z}_1^{\mathrm{r}}, \mathbf{z}_2^{\mathrm{r}})$	Inequality constraints evaluated at two-dimensional infeasible points
$\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{j}}\right)$	j^{th} constitutive constraint
$\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{X}, \boldsymbol{\theta}_{\mathrm{j}}\right)$	j^{th} constitutive constraint evaluated at feasible points
$\hat{\mathrm{g}}_{\mathrm{j}}\left(\mathbf{x}^{l}, \boldsymbol{\theta}_{\mathrm{j}}\right)$	j^{th} constitutive constraint evaluated at l^{th} feasible point
$\hat{\mathrm{g}}(\boldsymbol{\theta})$	Explicit constitutive constraint function identified
$\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Y}, \boldsymbol{\theta}_{\mathrm{j}})$	j^{th} constitutive constraint evaluated at infeasible points
$\hat{\mathrm{g}}_{\mathrm{j}}\left(\mathbf{y}^{k}, \boldsymbol{\theta}_{\mathrm{j}}\right)$	j^{th} constitutive constraint evaluated at k^{th} infeasible point
$\mathbf{g}^{*}(\boldsymbol{\beta})$	Updated vector of inequality constraints
J	Overall objective function
J_1	First part of the overall objective function
J_2	Second part of the overall objective function
J_3	Third part of the overall objective function
$\mathrm{KS}_{\mathrm{U}}(\mathbf{x}, \rho)$	Underestimator form of Kreisselmeier-Steinhauser function
$\overline{\mathrm{KS}}_{\mathrm{U}}\ (\mathbf{x}, \rho)$	Modified version of underestimator form of KS function

$KS_O(\mathbf{x}, \rho)$	Overestimator form of Kreisselmeier-Steinhauser function
$\overline{KS}_O(\mathbf{x}, \rho)$	Modified version of overestimator form of KS function
N	Number of sample points
N^b	Reduced sample size around boundaries
N^{Bnds}	Number of constitutive box constraints
N^{Circ}	Number of constitutive circular inequality constraints
N^{Con}	Number of total constitutive inequality constraints utilized
N^{dist}	Number of smallest distances
N^{Elps}	Number of constitutive ellipsoidal inequality constraints
N^f	Number of feasible points
N^{fb}	Number of sample points for boundary set of feasible points
N^i	Number of infeasible points
N^{ib}	Number of sample points for boundary set of infeasible points
N^{Lin}	Number of constitutive linear inequality constraints
N^{Type}	Number of definite types of form-specific constitutive constraints
$N^{\mathcal{H}}$	Number of neurons in hidden layer
$N^{\mathcal{O}}$	Number of neurons in output layer
NP	Population size
p	Parameter of percentile
$p_{\mathbf{X}}$	Parameter of percentile for set of feasible points
$p_{\mathbf{Y}}$	Parameter of percentile for set of infeasible points
$\mathbf{P}$	Sparse parameter matrix
P	Pressure
$\mathbf{R}$	Set of random sample points generated between zero and one
$\mathbf{T}$	Tensor of a Black-and-White image
t^k	Labelling value of each sample points for classification
$\boldsymbol{\mathcal{W}}^{\mathcal{H}}$	Weight matrix of hidden layer
$\boldsymbol{w}_i^{\mathcal{H}}$	Weight vector of hidden layer for i^{th} input neuron
$\boldsymbol{\mathcal{W}}^{\mathcal{O}}$	Weight matrix of output layer
$\boldsymbol{w}_m^{\mathcal{O}}$	Weight vector of output layer for m^{th} hidden neuron

$\mathbf{X}$	Set of feasible points
$\mathbf{X}^{b}$	Boundary set of feasible points
$\mathbf{x}^{l}$	l^{th} feasible point
$\mathbf{x}_{i}$	Column vector of i^{th} dimension of feasible points
$\mathbf{x}$	State variables
$\mathbf{Y}$	Set of infeasible points
$\mathbf{Y}^{b}$	Boundary set of infeasible points
$\mathbf{y}^{k}$	Row vector of k^{th} infeasible point
$\mathbf{y}_{i}$	Column vector of i^{th} dimension of infeasible points
$\mathbf{Z}$	Complete set of sample points
Z	Ethane mole fraction
Z^{LC}	Variable obtained by interpolation of saturated-liquid curve
Z^{VC}	Variable obtained by interpolation of saturated-vapor curve
$\mathbf{z}_{i}$	Column vector of i^{th} dimension of sample points
$\mathbf{z}^{L}$	Row vector of lower bound on optimization decision variables
$\mathbf{z}^{LB}$	Row vector of lower bound on sample points
$\mathbf{z}^{U}$	Row vector of upper bound on optimization decision variables
$\mathbf{z}^{UB}$	Row vector of upper bound on sample points
$\mathbf{z}_{1}^{b}$	Column vector of first dimension of feasible points in 2-D
$\mathbf{z}_{2}^{b}$	Column vector of second dimension of feasible points in 2-D
$\mathbf{z}_{1}^{r}$	Column vector of first dimension of infeasible points in 2-D
$\mathbf{z}_{2}^{r}$	Column vector of second dimension of infeasible points in 2-D
$\boldsymbol{\beta}$	Vector of process parameters (uncertain variables)
ε	Small error margin
μ	Weighting parameter of third sub-objective function
$\mathcal{N}_{ELM}(\mathbf{Z}, \boldsymbol{\theta})$	Neural constitutive constraint(s) generated via ELM
$\mathcal{N}_{NN}(\mathbf{Z}, \boldsymbol{\theta})$	Neural constitutive constraint(s) generated via NN
$\boldsymbol{\theta}$	Parameter vector of all constitutive constraints utilized
$\boldsymbol{\theta}_{j}$	Parameter vector of form-specific j^{th} constitutive constraint

$\boldsymbol{\theta}_{\mathrm{M}}^{-1}$	Inverse of square symmetric covariance parameter matrix
$\boldsymbol{\theta}^{\mathrm{I}}$	Parameter vector of form-free constitutive interaction constraint
$\boldsymbol{\theta}^{\mathrm{L}}$	Parameter vector of form-specific constitutive linear constraint
$\boldsymbol{\theta}^{\mathrm{LB}}$	Parameter vector of form-specific constitutive lower-bound constraint
$\boldsymbol{\theta}^{\mathrm{PQ}}$	Parameter vector of form-free constitutive pure quadratic constraint
$\boldsymbol{\theta}^{\mathrm{Q}}$	Parameter vector of form-free constitutive quadratic constraint
$\boldsymbol{\theta}^{\mathrm{UB}}$	Parameter vector of form-specific constitutive upper-bound constraint
ρ	Tightening parameter for Kreisselmeier-Steinhauser function
ρ_{d}	Density of the mixture
$\boldsymbol{\Phi}(\mathbf{Z})$	Augmented design matrix
$\boldsymbol{\Phi}^{\mathrm{I}}(\mathbf{Z})$	Design matrix of constitutive linear-interaction constraint
$\boldsymbol{\Phi}^{\mathrm{G}}(\mathbf{Z})$	Generic design matrix
$\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})$	Design matrix of constitutive linear constraint
$\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_{\mathrm{i}})$	Design matrix of constitutive lower-bound constraint
$\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})$	Design matrix of constitutive pure quadratic constraint
$\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_{\mathrm{i}})$	Design matrix of constitutive upper-bound constraint
$\boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{Z})$	Design matrix of constitutive quadratic constraint
$\|.\|_1$	1-norm operator
$\varphi(.)$	Activation function operator
$\mathbb{C}(.)$	Constraint aggregation operator
$\mathrm{dist}\{.\}$	Distance operator
$\min\{.\}$	Minimum operator
$\mathrm{U}(0,1)$	Uniform random number generation operator between zero and one
$\mathrm{vec}(.)$	Vectorization operator
$\mathbb{S}\{.\}$	Distance-sorting Operator
$\odot$	Hadamard product

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
BW	Black-and-White
CA	Constraint Aggregation
CAD	Cylindrical Algebraic Decomposition
CART	Classification and Regression Trees
ChE	Chemical Engineering
CMAES	Covariance Matrix Adaptation Evolution Strategy
CNN	Convolutional Neural Networks
CPU	Central Process Unit
CSTR	Continuous Stirred-Tank Reactor
DE	Differential Evolution
DIRECT	Dividing hyper-RECTangle
DNN	Deep Neural Network
DT	Decision Trees
ELMs	Extreme Learning Machine
GAN	Generative Adversarial Networks
GPR	Gaussian Process Regression
IE	Induced Exponential
IP	Induced Power
k-NN	k-Nearest Neighbours
KS	Kresselmeier-Steinhauser
LP	Linear Programming
MCS	Monte Carlo Sampling
MINLP	Mixed Integer Nonlinear Programming
ML	Machine Learning
MS	Meshwise Sampling
NLP	Nonlinear Programming

NNs	Neural Network
PNN	Probabilistic Neural Network
QP	Quadratic Programming
ReLU	Rectified Linear Unit
ReSech	Rectified Hyperbolic Secant
RS	Random Sampling
RSLFNN	Random Single Layer Feedforward Neural Network
SLFNN	Single Layer Feedforward Neural Network
SLHS	Stratified Latin Hypercube Sampling
SVM	Support Vector Machine
WOP	Williams-Otto Plant

1. INTRODUCTION

The determination of feasible operation regions of chemical processes is one of the important topics in process system engineering. In this context, there are many studies in the literature. Some of them focus on the determination of flexible operation region within a feasible region formed by known inequalities, while others attempt to identify a feasible region as a black-box formed by unknown inequalities. Additionally, there exist studies that re-consider some engineering and image-processing applications from optimization perspective. We begin this chapter by introducing the major and minor goals of this thesis work and continue by a short summary of main contributions of the literature works related to the topics of this thesis work together with a brief review of them. We close this chapter by presenting the outline for the rest of this thesis work.

Major goal of the data-based-constraint-identification task is to form a unique optimization formulation that can pave the way for the identification of D-dimensional feasible/infeasible regions, whatever the types feasible or infeasible regions are, i.e., joint, disjoint, and fragmented convex or nonconvex regions. This task can be handled with two approaches; via explicit and implicit mathematical equations, i.e., either by using form-specific and form-free explicit constitutives, and via neural constitutives, i.e., by using form-free implicit neural constitutives. Both of these two approaches must rely on "constraint aggregation" that provides single-explicit and single-implicit mathematical equations capable of describing joint or disjoint single or multiple regions. The constraint aggregation when fused with the above mentioned two alternative approaches delivers two tools: "single high-order explicit equation-based constraints" and "single neural constitutive constraints". Other minor goals of this thesis work are to study the applicability of some classification tools for the constraint identification, e.g., for the periphery identification of feasible regions, and to demonstrate that our algorithm can be deployed within the topic of image processing, e.g., for the generation of image-based constraints.

The works in the literature focus mostly on the determination of flexible-operation regions and flexibility index of feasible-operation regions. Studies that attempt to identify

feasible regions formed by implicit (unknown) inequality constraints depend generally on the classification of feasible and infeasible points by utilizing ML algorithms as surrogate models and do not promise the identification of feasible regions with single or multiple explicit mathematical formulations. Even though there exist some studies dealing with determination of flexible regions of chemical processes with explicit symbolic formulations, applications of such studies are generally confined to low-dimensional problems. Additionally, most of the studies do not provide identification cases of disjoint regions further apart from each other and examine only identification of single or contiguous regions.

Below, starting with the year 1999, we briefly review the literature works related to the thesis topic, in chronological order.

Morito et al. (1999) focused on developing an iterative approach that determines the constraints from the simulation results of the designs of logistic systems and embedded the determined constraints into optimization of the logistic systems.

Ierapetritou (2001) worked on development of a method accommodating the convex-hull algorithm for identification of convex feasible regions and determination of their flexibility index. However, the proposed technique is not suitable for nonconvex feasible regions.

Banerjee and Ierapetritou (2002) developed a novel approach which maps relationship between input and output variables by considering flexibility and design optimality of systems and gave a parametric representation of optimum objective consisting of uncertain parameters only.

Benko et al. (2002) provided a numerical approach to suit multiple curves and multiple surfaces to three-dimensional data obtained from reverse engineering by considering them as equality constraints.

Goyal and Ierapetritou (2002) proposed a novel method, relying on the notion of inner and outer approximation of feasible regions to determine ranges of operating region

where the points included in the operation region are not only feasible, but also gives the most profitable results considering objective function of the optimization problem. With this study, they aimed to remedy underestimation issue of feasible regions.

Dabbene et al. (2003) developed two algorithms for inward approximation of feasible regions formed by several linear inequalities in n-D via single ellipsoidal inequality constraint. The first algorithm attempts to find an ellipsoid inside the D-dimensional feasible regions in accordance with its predefined size. The second one relies on finding the greatest ellipsoid inside the feasible regions in n-D.

Goyal and Ierapetritou (2003a) extended their previous study (Goyal and Ierapetritou, 2002) to identification of nonconvex feasible operation regions. By this work, they claimed that it was possible to identify feasible operation regions, even if they are inherently nonconvex, with outer approximation to determine infeasible zones, and then, with simplicial approximation to identify expanded feasible regions built through discarding constraints which make the feasible region nonconvex. However, to utilize this proposed framework for identification of nonconvex feasible region, source of nonconvexity must be known.

Goyal and Ierapetritou (2003b) developed a systematic approach to provide a collection of optimal process designs for different market requirements corresponding to different ranges of data by combining data analysis and optimization of process design. This approach relies mainly on clustering technique utilized.

Sirdeshpande et al. (2005) worked on finding optimal configurations of air-separation cycles via modelling them algebraically, which eventually resulted in MINLP formulation. This study covered the determination of flexibility index for the optimal configurations of the processes as well.

Benjelloun et al. (2007) developed an approach which resorts to polynomial fitting to complete incomplete/open contours obtained from images (which result in erroneous determination of image boundaries) via edge detection tools of image processing.

Liu et al. (2008) developed an algorithm which relies on delaunay triangulation as boundary detection algorithm by viewing the clustering technique from a statistical perspective.

Banerjee et al. (2010) developed a strategy for feasibility check, which considers representation of original processes with approximate black-box systems, not only by preserving original complexity of processes in modelling stage but also by reducing computational cost.

Boukouvala et al. (2011) provided a novel approach that uses kriging to analyze feasibility of processes which cannot be explicitly formulated, and a method of adaptive sampling to reduce cost of sampling. This approach requires the initial sampling points which properly represent the feasible region.

Jeong et al. (2012) employed Support Vector Machine (SVM) algorithm to predict feasibility of unseen design points. This study included utilization of k-nearest neighbour algorithm to reduce number of training points as well, which resulted in shorter training times for the SVM algorithm. Additionally, other studies to identify feasible regions by utilizing a classifier algorithm such as SVMs and probabilistic neural network exist in the literature (Basudhar et al., 2008, 2012; Basudhar and Missoum, 2010; Fang et al., 2022; Patel and Choi, 2012; Singh et al., 2017).

G. Wu (2015) developed an image-boundary (edge-detection) algorithm incorporated with tangent line and ellipse fitting for rapid prototyping technology.

Kovács et al. (2015) developed a method of constraint fitting for the application of reverse engineering based on the previous work of Benko et al. (2002). The proposed approach aims to remedy some issues related to the creation of the models of Computer-Aided Design from estimated data in digital environment.

Adi et al. (2016) introduced a method to compute volumetric flexibility index of a chemical system. This novel approach includes utilization of random-search algorithm to generate feasible points used to accurately obtain boundaries of feasible regions, and

implementation of delaunay triangulation to form simplexes relying on generated feasible points close to boundaries. They proclaim that this novel approach works for high-dimensional and nonconvex regions as well.

Astorino et al. (2016) suggested the use of generic formulations of some probable shapes, such as ellipsoidal, spherical, thereby achieving not only nonlinear separation of two classes, but also explicit mathematical equations of boundaries, which cannot be extracted via conventional SVMs due to transformation from natural domain to a kernel-function domain. Additionally, they propose a novel semi-supervised technique for data points with unknown labels.

Zhang et al. (2016) worked on development of a surrogate model designed by the concept of mixed-integer linear constraints resulting in determination of feasible regions with the union of multiple convex regions. They claimed that their linearly-formulated surrogate model can accurately and efficiently approximate convex or nonconvex feasible regions and objective function of optimization problem by taking advantage of an existing set of data points and scalar values of objective function corresponding to each of these data points.

Chen and Fuge (2017) devoted their study to efficiently identify feasible regions, especially fragmented or disjoint feasible regions, in presence of implicit inequality constraints. By the active-learning method proposed in this work, they aim to determine feasible regions by imposing box constraints on design variables, which will expand towards ultimate identification of feasible regions.

Na et al. (2017) modified the Dividing hyper-RECTangle (DIRECT) algorithm by empowering DIRECT algorithm with a sub-dividing step which allowed the algorithm to take implicit constraint into account during optimization of chemical processes simulated via commercial process simulator. In this work, optimization of cryogenic refrigerant process is executed by the proposed algorithm with many different optimization techniques. By comparing specific power necessary for natural-gas liquefaction they demonstrate effectiveness of their proposed algorithm.

Zhao and Chen (2018) worked on determination of flexibility region via Cylindrical Algebraic Decomposition (CAD) technique in presence of known inequality constraints. This work is committed to describing flexibility region with explicit functions which depend both on uncertain and control variables. They make their method applicable to a system described by nonpolynomial terms, since CAD technique is only utilized for polynomial functions, via polynomial fitting. However, the technique proposed in this work is suitable for determination of flexibility index of low-dimensional regions, since CAD requires symbolic computation and such computational procedure is too expensive for high-dimensional input space.

Knudde et al. (2019) focused on accurate modeling of feasible regions obtained via computationally-expensive simulations by utilizing active-learning technique. This approach promises discovery of boundaries of regions rather than whole regions with reduced sample points.

Kusumo et al. (2019) centred on development of a nested sampling strategy, relying on probabilistic estimation of the feasibility of sample points, to better comprehend design spaces of pharmaceutical processes.

Zhao et al. (2019) proposed a novel method which utilizes CAD technique for projection of solution space onto one dimensional space for determination of flexibility indexes of process. As opposed to many studies interested in process flexibility, they claim that success of their novel approach is invariant to convexity of feasible regions. However, the applications of such novel approaches accommodating CAD is limited to low-dimensional cases due to computational burden.

Al et al. (2020) constructed a Monte-Carlo based systematic approach which combined physics-based process simulations with some tools used for assessment of uncertainty of systems such as ML algorithms, Sobol sensitivity analysis, Monte Carlo simulation for process synthesis and determination of optimal configurations of complex process systems. This methodology covers a technique which can model each constraint separately for maintaining feasibility as well.

Hu and Qu (2020) provided a comprehensive study to examine how generalized constraints, regarded as implicit or black-box constraints of the system, influence the creation of ML-based systems.

Kovács and Várady (2020) presented a methodology to reconstruct engineering components represented by measured data in the context of reverse engineering. In this study, they emphasize the need for imposing geometric constraints in order to find optimum control points of B-spline curves for free-form shapes.

Kucherenko et al. (2020) proposed a novel procedure for identification of probabilistic design space by meta modelling and adaptive sampling. This study aims to remedy the computational burden of model-based practices necessitating simulations through all certain and uncertain parameters of processes.

Metta et al. (2020) aimed at identifying feasible regions, whose inequalities cannot explicitly be obtained from process models, via a novel method combining ANN-based surrogate model with an adaptive-sampling method executed by means of jack-knifing. They claim that this method works better for determination of high-dimensional feasible regions compared to previous studies utilizing a surrogate model built via the kriging method.

Shi et al. (2020) introduced a novel technique of boundary sampling for time-dependent reliability-based design optimizations. By this sampling technique, surrogate-based method utilizes feasible and infeasible points near boundaries and excludes infeasible points far away from boundaries, while the similar works in the literature focus on development of surrogate models of constraints in feasible and infeasible regions by taking all points into account.

Qing et al. (2020) contributed to the studies which use kriging as surrogate model by adding a local penalization function, which relies on the assumption that constraint function is Lipschitz continuous and takes the impact of batch samples selected for a certain iteration into account.

Zheng et al. (2020) claimed that their novel method, accommodating CAD technique, can be applied to high-dimensional cases as opposed to other works attempting to utilize CAD technique to obtain symbolic (explicit) representations of flexible region. However, explicit representations of flexible region (inequalities) consist of only two uncertain variables and relies on elimination of state variables through equalities.

Ghobadi and Mahmoudzadeh (2021) develop an approach to obtain best value of implicit objective function, while satisfying black-box linear inequality constraints forming a convex feasible region by utilizing inverse optimization and feasible points given by an expert.

Straus et al. (2021) attempted to decrease number of sampling points by limiting sampling zones via creation of additional linear inequalities formed by taking interdependencies between variables into account. They build a surrogate model based on penalized regression and error-maximization sampling subject to additional linear constraints. They include an application of their proposed methodology to hydrogen production.

Wu et al. (2021) were interested in improving accuracy and efficiency of kriging-based method for use of reliability-based design optimizations for probabilistic constraint problems, and they proposed “probability feasible region enhanced importance boundary sampling” (PFRE-IBS) method.

Badejo and Ierapetritou (2022) provided a complete framework for supply-chain operations and scheduling problems, and eventually formulated supply-chain operations and scheduling as multi-objective optimization ones. They utilize SVM algorithm and some linear models as surrogate constraint for the feasibility check.

Kumar et al. (2022) presented COUNT-CP algorithm that utilizes basic grammars and can give generalized form of constraints for prediction of unseen data in the context of constraint programming.

Ludl et al. (2022) studied exploration of design spaces of chemical processes via ML-based approach by considering converged points obtained via flowsheet simulation together with additional inequality constraints.

Zhang et al. (2022) utilized a Deep Neural Network (DNN) as classifier for surrogate constraint to learn feasible region from simulation results and predict whether new sampling points are feasible or not. After this, they trained another DNN as a regressor for surrogate model of objective function using sample points predicted as feasible together with values of objective function evaluated at these feasible sample points. They proclaim that this methodology efficiently works for high-dimensional problems even with small-sized training sets.

Nagar et al. (2022) proposed a methodology covering interpretable self organizing maps in order to resolve visualization issues of n-dimensional feasible regions.

For determination of operational flexibility based on volumetric flexibility index, Wang et al. (2022) worked on developing a novel method which directs adaptive sampling towards boundaries of regions in order to reduce sampling cost.

Zhao et al. (2022) focused on building a novel approach which aims to obtain symbolic description of flexible operation regions (feasible regions denoted only with uncertain parameters) by combining the work of Zhao et al. (2019) with two sampling procedures. They introduce an adaptive sampling method designed as an optimization problem depending on initial feasible points and the KS aggregation function and propose a line-projection technique for generation of sample points at boundaries. Polynomial fitting is utilized to find a mathematical equation of the boundary represented by KS aggregation function. After obtaining this polynomial representation, they utilize CAD method to convert polynomial surrogate model into series of sub-equations of the boundary.

Zhao, Ochoa, et al. (2022) focused on determination of flexibility index via bi-level optimization together with generic formulations of specific shapes including ellipse and

diamond. This work covers development of a derivative-free optimization algorithm to solve nonconvex optimization problems as well.

After the review of the literature works, we give the outline of this thesis work chapter by chapter as follows:

In Chapter 2, we build comprehension of the feasible and infeasible regions and discuss the types of such regions in the context of constraint-identification task.

In Chapter 3, we provide three sampling methods utilized in the thesis work. These are Random Sampling (RS), Stratified Latin Hypercube Sampling (SLHS), and Meshwise Sampling (MS). Together with the explanation of our sampling procedure we discuss the effects of sampling methods on the constraint-identification task. We introduce the concept of boundary-zone formation as well.

In Chapter 4, we explain the technique of constraint aggregation and provide some information about the Kreisselmeier-Steinhauser function. We discuss why the task of constraint identification requires the use of an aggregation function as well.

In Chapter 5, we establish the first concept of the constitutive constraints as “form-specific constitutives”, and thoroughly elucidate the basis of our algorithm, including the terms in the objective function, step by step.

In Chapter 6, we present constraint-identification examples of convex feasible regions solved via the form-specific constitutives as explained in Chapter 5. This chapter can be deemed as the test chapter showing that our algorithm with the form-specific constitutives and its overall objective function over the boundary sample points work as conceived.

In Chapter 7, we develop the second concept of the constitutive constraints as “form-free constitutives”, and re-develop some of the form-specific constitutives via the design-matrix approach. Through this chapter, we attempt to bring our algorithm to the next level in terms of its capability to identify more complex regions by equipping it with a new

concept of constitutives without any modification or change in the overall objective function. By this chapter, we discuss the other possibilities for the construction of novel concepts of constitutives as well. This chapter also heralds the establishment of a novel concept of constitutives via Machine Learning (ML) algorithms.

In Chapter 8, we present constraint-identification examples of nonconvex feasible regions solved with different combinations of the form-specific and form-free constitutives and with single higher-order polynomial constitutive constraints, all of which are built via the technique of design matrix. This chapter accommodates not only the examples of exact identification of fragmented and disjoint regions, but also a particular example of approximate identification of a convex-circular feasible region as well.

In Chapter 9, we establish the third concept of constitutive constraints as "neural (machine-learned) constitutives", by the use of some ML algorithms such as NN and ELM, thereby making our algorithm reach the peak of its identification capability of feasible/infeasible regions. We present the constraint-identification examples of two different feasible regions (one is convex, while the other is nonconvex) learned via neural constitutives within seven different cases that provide different utilization of the ML algorithms (as single output or multiple output NNs/ELMs) with different activation functions. This chapter presents an unorthodox activation function newly created for this thesis work as well.

In Chapter 10, by leaving our algorithm aside, we elucidate how to identify the peripheries of feasible regions by the classification of feasible and infeasible points via the utilization of some classification algorithms including ML-based ones such as Probabilistic Neural Network (PNN), k-Nearest Neighbour (k-NN), Support Vector Machine (SVM), Gaussian Process Regression (GPR), and Classification and Regression Trees (CART). In this chapter we also scrutinize the effects of one of the hyperparameters of these algorithms on the periphery identification of feasible regions through some examples.

In Chapter 11, we discuss the idea of whether it is possible to attain the mathematical equations of boundaries of a Black and White (BW) image. Towards the achievement of this idea, we suggest the concept of "image-based constraints" which relies on the

transformation of a BW image into a region formed by feasible and infeasible points. We support the idea of image-based constraints with five cases encompassing the identification of the boundaries (constraints) of four BW images named as “droplet”, “V-shaped cup”, “disjoint regions”, and “human”.

In Chapter 12, we exemplify the application of the work of “data-based constraint identification” to chemical engineering examples including the identification of feasible-operation regions of a Continuous Stirred-Tank Reactor, the Williams-Otto Plant, and a phase-equilibrium envelope (with and without the existence of uncertainty). We utilize multiple-explicit constitutive constraints for the examples of the identification of feasible-operation regions and deploy single 3^{rd}-order polynomial constitutive constraints for the example of phase equilibrium envelopes. Additionally, we generate feasible and infeasible points relying on experimental data, with and without measurement error (under uncertainty), for the examples of phase equilibrium envelopes.

In Chapter 13, which is “Conclusions and Recommendations”, we summarize the work of “data-based constraint identification” chapter by chapter and put forward some ideas which can be considered for the future studies.

We build algorithm of “data-based constraint identification” using MATLAB and provide some sample codes of our algorithm in the Appendix. Lastly, it should be highlighted that two-dimensional aggregated constraints and individual-implicit constitutive constraints are all depicted via MATLAB’s “contour” function. Two-dimensional individual-explicit constitutive constraints are drawn via MATLAB’s “fimplicit” function. For representations of three-dimensional aggregated constraints and individual constitutive constraints, MATLAB’s “isosurface” and “fimplicit3” functions are utilized, respectively.

2. THE FEASIBLE REGION

Plant operations may inherently have certain limitations. The freedom of selection of the plant's operating points may be restrained by plant limitations which are usually due to design and/or thermodynamic restrictions. Operating variables that can provide optimal operation of the plant within plant limitations may be selected as optimization decision variables. Optimization decision variables are not necessarily chosen from the set of operating variables, they can be functions of all or some of the operating variables. Not all operating variables may be qualified as optimization decision variables for optimal plant operations. The change in operating variables must be confined to feasible operating region. Implicit or explicit functions of operating points (or, operating variables) that describe plant's limitations are called "constraints". Thus, it is important to determine feasible operating regions properly in terms of constraints which are functions of decision variables and/or operating variables. Inequality constraints specify the maximum or minimum extent to which operating points can change without violating the plant limitations. Detecting feasible regions formed by constraints is not only at the centre of this thesis work but also is an important task to pursue safe and economically-optimized plant operations. Therefore, the subject is of interest to researchers as well as to process engineers.

The feasible region emerges when all inequality constraints are considered together. Mathematical functions describing the constraints determine if the feasible region is convex or nonconvex. To test the convexity of feasible region, the constraint functions must be individually tested in terms of convexity or concavity. If the feasible region is convex, then all of the inequality constraint functions (which depend on decision variables or operating points, $\mathbf{z}$) forming the region, expressed in the form of $\mathbf{g}(\mathbf{z}) \leq \mathbf{0}$, must be individually convex. Nonlinear constraint functions can be convex or concave, whereas linear constraints prove to be the both.

Some technical explanation about how the constraint function is regarded as convex or concave may be given here. The sign of the second derivative of the constraint function, if the function is twice-differentiable with respect to variables, determines its convexity. If

the second derivative of the constraint function is positive, then the constraint is convex. If the second derivative of the constraint function is negative, then the constraint is concave. For multivariable convex constraint functions, all the eigenvalues of its second derivative matrix, called the Hessian matrix, must be non-negative. If the multivariable function has a quadratic dependence on its variables, then its Hessian matrix is a constant matrix and thus the convexity test result is valid globally, i.e., for any values of the variables. However, for general nonlinearity, the Hessian matrix is dependent on variables and thus the convexity test is valid only at the current values of the variables. Obviously, all multivariable linear constraint functions are convex.

Therefore, since linear equations are both convex and concave, the feasible region formed only by the linear constraints becomes convex. In other words, the presence of only linear constraints guarantees the convexity of the feasible region.

After recapping the knowledge about the convexity of the feasible region and constraint functions, we can employ an illustrative set of constraints forming a nonconvex feasible region to better understand that a nonconvex region cannot be formed by only linear constraints. It should be noted that if the number of variables is greater or equal to four, it is impossible to depict the feasible region. For convenience, the feasible region including two variables is utilized to examine the properties of the feasible operating region. This demonstrative set of constraints are given below:

$$g_1: \quad +z_1 - 7 \leq 0, \tag{2.1a}$$

$$g_2: \quad -z_1 + 1 \leq 0, \tag{2.1b}$$

$$g_3: \quad +z_2 - 5 \leq 0, \tag{2.1c}$$

$$g_4: \quad -z_2 \leq 0, \tag{2.1d}$$

$$g_5: \quad (z_1 - 0.5)^2 + (z_2 - 0.5)^2 - 8 \geq 0. \tag{2.1e}$$

Figure 2.1 demonstrates the convex feasible region formed by the first four linear constraints given above by omitting the last circular inequality constraint. The filled blue points belong to the feasible set of trial points $\left(\mathbf{z}_1^{\mathrm{b}}, \mathbf{z}_2^{\mathrm{b}}\right)$ satisfying the four linear inequality constraints, i.e., $\mathbf{g}\left(\mathbf{z}_1^{\mathrm{b}}, \mathbf{z}_2^{\mathrm{b}}\right) \leq \mathbf{0}$, and the hallow red points belong to the infeasible set of trial points $(\mathbf{z}_1^{\mathrm{r}}, \mathbf{z}_2^{\mathrm{r}})$ violating one or more inequality constraints, i.e., $\mathbf{g}(\mathbf{z}_1^{\mathrm{r}}, \mathbf{z}_2^{\mathrm{r}}) > \mathbf{0}$. The arrows show the feasible direction of the constraints. As can be seen, the feasible (blue)

region is the intersection of the feasible directions of the individual inequality constraints. From now on, the feasible direction arrows will mostly be omitted for the sake of clarity since after this point it should be obvious that the region of blue points is the feasible region and its outside, i.e., the region of red points, is the infeasible region.

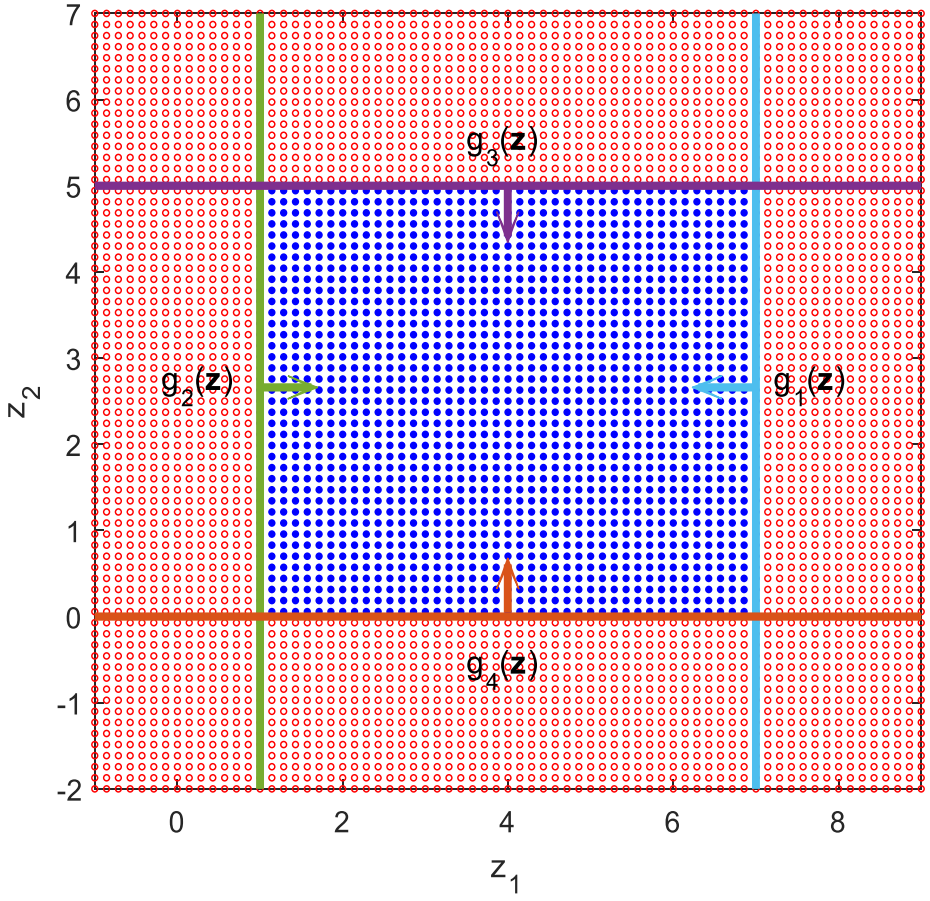

Figure 2.1. An illustrative two-dimensional convex feasible region formed by four linear inequality constraints.

Figure 2.2 shows a nonconvex feasible region, this time, consisting of all five inequality constraints given above in Equation (2.1). Although the first four constraints are linear, the last constraint, the circular constraint, is nonlinear. The nonconvexity of this region is due to this last circular constraint since the feasible direction is of $g(z_1, z_2) \geq 0$ type and in this direction the constraint is concave.

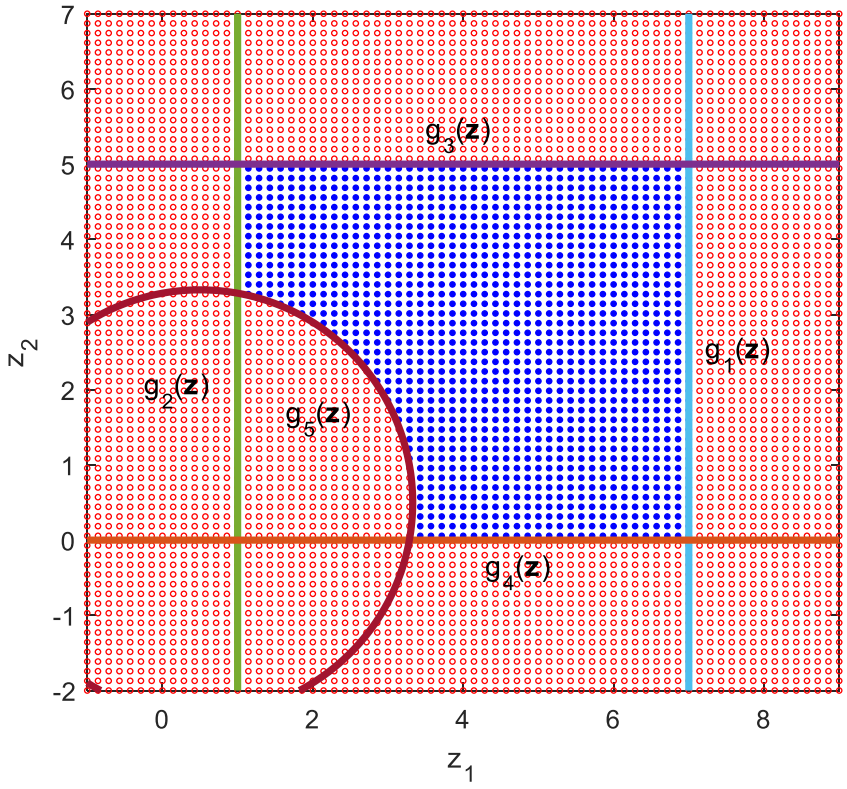

Figure 2.2. An illustrative two-dimensional nonconvex feasible region formed by four linear inequality constraints and one circular constraint.

In the constraint identification task (which will be introduced in Chapter 5), actual constraints are unknown or obscure. Thus, if all the regressor-constraint functions (which will be introduced in Chapter 5 as well) are selected to be linear, the nonconvex feasible region, such as the one depicted above, cannot be surrounded. This is true even if large number of linear regressor constraints are used. Therefore, at least one of the constraint functions taken into the set of regressor constraints should have the capability to capture nonlinear / nonconvex constraints. This can be explained with the following set of figures, Figure 2.3 and Figure 2.4.

Figure 2.3 is a false representation of a hypothetical nonconvex feasible region formed via nonconvex penetration of two linear constraints. In this figure, the curvature of the nonconvex circular constraint is approximated by two linear inequality constraints. At first, to the novice eye, it may be seen that there is nothing wrong with the depicted situation. The novice eye may think that the approximation can be improved more and

more by the use of increasing number of such linear inequality constraints. In other words, it may be thought that the nonconvex feasible region indicated by the set of blue points can be well surrounded by using only linear inequality constraints, many of which are employed for the nonconvex penetration.

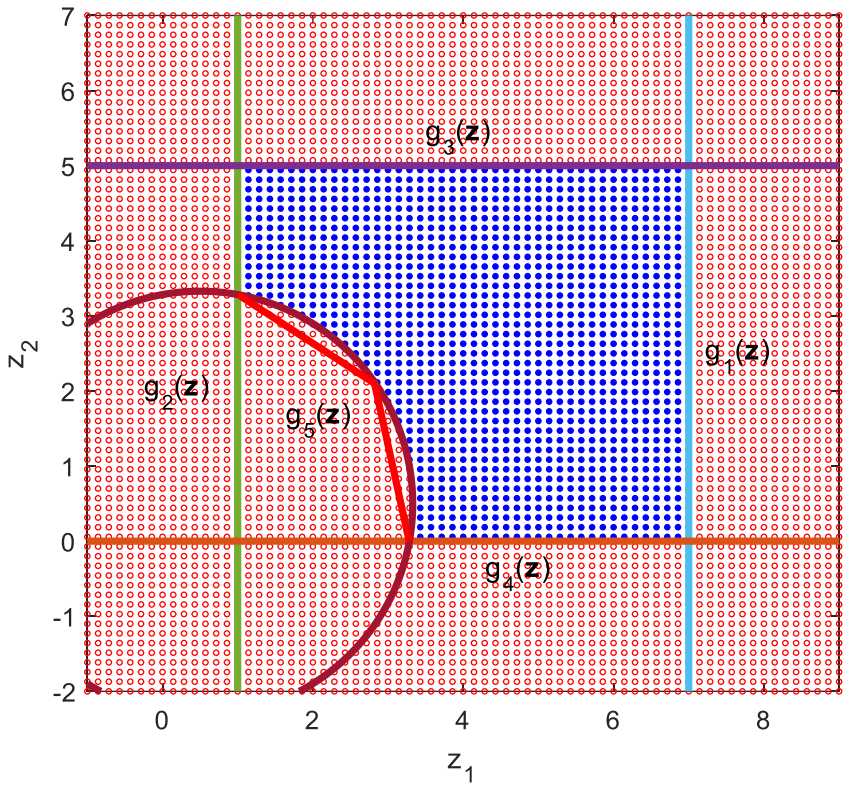

Figure 2.3. A false representation of a hypothetical feasible region formed via nonconvex penetration of linear constraints.

However, the above-mentioned representation in Figure 2.3 is theoretically impossible as demonstrated by Figure 2.4 below. If one draws the linear inequality constraints penetrating the nonconvex region with full length together with the arrows showing the feasible direction of each one of them and looks at the union of the feasible directions of all the linear inequality constraints, the true feasible region that can be surrounded by linear constraints becomes obvious, which is the region surrounded by the thick black lines. This true region of the linear inequalities obviously cannot surround the nonconvex circular penetration. Therefore, theoretically, the situation depicted in Figure 2.3 is totally impossible.

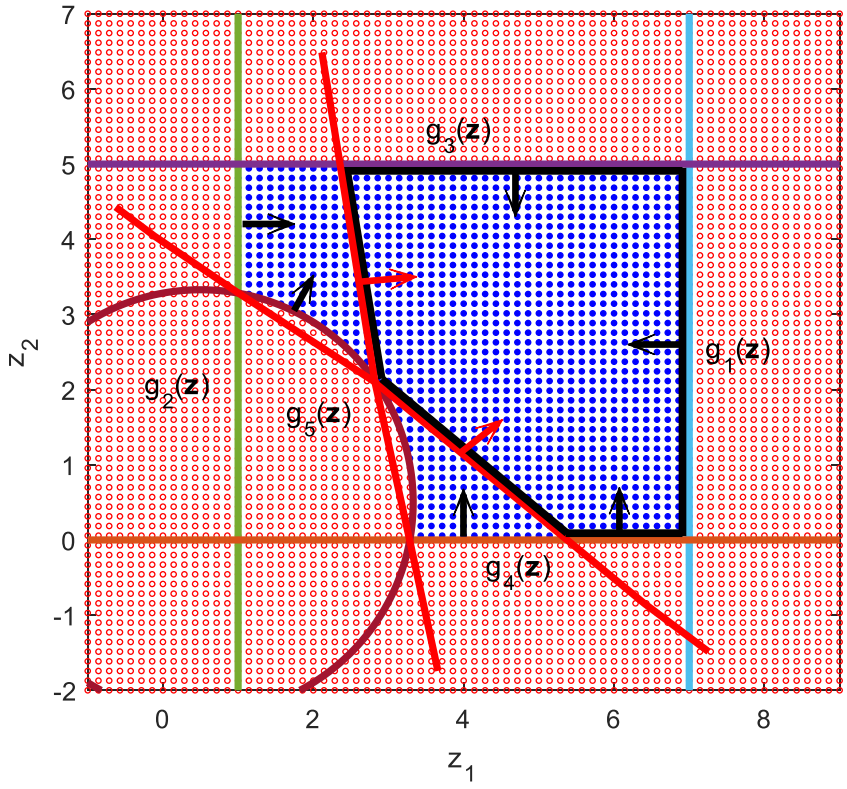

Figure 2.4. The true representation of the hypothetical nonconvex feasible region only via linear constraints.

One question concerning the constraint identification task (which will be introduced in Chapter 5) may arise when these figures are further scrutinized. Is it possible to achieve constraint identification (to tightly surround the feasible region via regressor constraint functions) by considering the regressor constraints individually, one-by-one? To answer this question, it is beneficial to see how each one of the regressor constraints alone delineates the feasible region by looking at the feasible points in the focus of its own feasible direction.

Figure 2.5a and Figure 2.5b show feasible regions in terms of only one constraint's feasible direction, which are $g_1(\mathbf{z})$ and $g_5(\mathbf{z})$, respectively. In Figure 2.5a it is seen that with regard to only linear inequality constraint, $g_1(\mathbf{z})$, all points to the left of the constraint line are feasible. Some of these points feasible to the linear inequality constraint, however, fall in the infeasible region of the circular inequality constraint in Figure 2.5b. Similarly, in Figure 2.5b, it is seen that with regard to only circular inequality constraint, $g_5(\mathbf{z})$, all

points to the right (northeast) of the circular constraint are feasible. However, again, some of these points feasible to the circular inequality constraint, fall in the infeasible region (to the right) of the linear inequality constraint in Figure 2.5a.

Obviously, the actual feasible region is composed of the intersecting set of points that are feasible to both of the inequality constraints jointly. Therefore, it is not possible to distinguish the conjoint feasible points from infeasible ones by considering the inequality constraints mutually exclusively. In other words, the labelling of a particular point (z_1, z_2) as feasible or infeasible cannot be done constraint by constraint but must be done considering all constraints simultaneously. Thus, the constraint-identification task requires a mathematical aggregation of all inequality constraints that merge their feasible directions into a joint feasible region. This critical mathematical tool is the KS constraint aggregation function which will be introduced in Chapter 4. Its application to constraint identification will be discussed in Chapter 5.

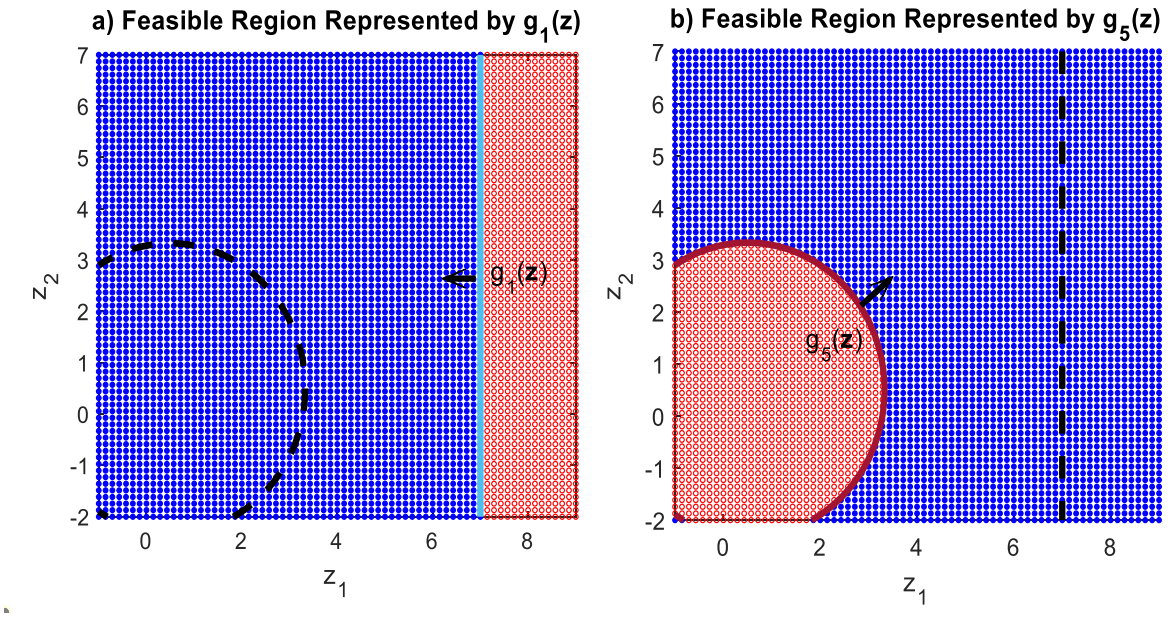

Figure 2.5. Individual feasible regions of two inequality constraints.

3. FEASIBLE REGION GENERATION AND SAMPLE SIZE REDUCTION

In this part, some sampling methods will be explained and compared to determine the best sampling method in the context of the constraint-identification task.

Firstly, in Section 3.1, titled “Generation of Feasible Region via Constraint Sampling”, we begin with elucidating some sampling methods, which are simple Random Sampling (RS), Stratified Latin Hypercube Sampling (SLHS), and Meshwise Sampling (MS). After brief explanations of these methods, we provide some notations which direct readers through the entirety of this thesis work. Then, we complete this section with the comparison of these sampling methods.

Lastly, in Section 3.2, titled “Sample Size Reduction via Boundary Zone Formation”, we propose the utilization of the reduced-size samples to form boundaries of the regions used in Section 3.1 with the purpose of cutting the computational load down on the optimizer during constraint identification. Then, sample-size reduction is thoroughly discussed.

3.1. Generation of Feasible Region via Constraint Sampling

Random Sampling (RS) is the first method used to produce sample points between specified lower and upper values. This method is also called Monte Carlo Sampling (MCS) (Shields and Zhang, 2016). MCS relies completely on random actions, and it is hoped that its random actions add up to space-filling. However, there are some disadvantages of this method due to its uneven randomness. For instance, this method may cause clusters of points in certain parts of the region, which, in turn, makes this region under-represented (Garud et al., 2017). Many different methods to resolve this issue are proposed by researchers, one of which is the Stratified Latin Hypercube Sampling (SLHS) that fills the space with much less uneven randomness compared to RS. Thus, we utilize SLHS as well for producing sample points.

SLHS differs from RS in that the former divides each dimension into K equivalent strata which have $1/K$ edge length. As a result of this division of dimensions, if there exists D-dimensional space, K^D strata arise. The salient property of this method is that each stratum takes only one sample point, which means that it remembers where the previous sample points are assigned (Garud et al., 2017). Like RS, SLHS randomly generates points between a specified lower and upper values, and thus the strata should be determined with the lower and upper bounds (Shields and Zhang, 2016). Although SLHS also accommodates randomness, the stratified assignment of sample points alleviates shortcomings of RS caused by uneven randomness to some extent. Thus, it is expected that SLHS provides a better-sampled (better space-filled) region than simple RS.

The last sampling method used in this thesis work is Meshwise Sampling (MS) which fills a region with the uniformly-sampled sample points over the grid of Cartesian coordinates. MS is the most basic sampling method compared to other sampling methods. One of the disadvantages of this method is that sample size exponentially increases with increasing dimension if sample size is fixed for each dimension, especially with small grid size. Thus, it is more suitable for problems that have small dimensions typically less than or equal to three if the sample size (grid size) is unchangeable for each dimension (Garud et al., 2017). However, when considered for the constraint-identification task, this method may be superior to other sampling methods in revealing the feasible-region boundary correctly, especially at the corners, even with fewer sample points due to regular distribution of sample points including those at the corners. We will elucidate the doubts on which method has more advantages in constraint-identification task by illustrating these methods with figures as well.

Before scrutinizing the effects of sampling methods and the number of sample points by depicting the regions with different sampling methods and different number of samples, we can set some notations for constraint identification as well, based on how sampling is executed in the context of this work. It should be also noted that these notations, created step by step during this chapter, are valid for the rest of this thesis work as well.

The first parameter which should be specified before sampling is the cardinality of the set of sample points, N. It is important to select the total number of sample points

suitably since the constraint-identification algorithm cannot determine inequality constraints accurately if the number of samples is not high enough to capture the feasible-region boundary closely, which is called under-sampling. Unlike under-sampling, over-sampling emerges where the sample points are needlessly excessive, which, in turn, adds unnecessary computational load to the optimizer. The selection of the cardinality of the sample set including all points in tandem with over-sampling and under-sampling will also be illustrated and further discussed by depicting these situations.

With RS and SLHS, all N points in D dimensions are generated randomly between zero and one using uniform random numbers, $U(0,1)$, and with MS, all N points in D dimensions are generated uniformly between zero and one, for each coordinate direction d, $d = 1, \dots, D$. The complete set of such generated points is called $\mathbf{R}$, where $\mathbf{R}$ is $N \times D$ matrix, the elements of which are between zero and one (non-inclusive for RS and SLHS, inclusive for MS).

It is vital to determine the lower and upper bounds on sample points properly since the range of sample points should completely cover the range in which both the feasible and infeasible regions are included. The lower and upper bounds of variables in each coordinate are packed as a row vector, which can be written as:

$$\mathbf{z}^{\mathrm{LB}} = [z_1^{\mathrm{LB}} \quad z_2^{\mathrm{LB}} \cdots \cdots z_{D-1}^{\mathrm{LB}} \quad z_D^{\mathrm{LB}}], \tag{3.1a}$$

$$\mathbf{z}^{\mathrm{UB}} = [z_1^{\mathrm{UB}} \quad z_2^{\mathrm{UB}} \cdots \cdots z_{D-1}^{\mathrm{UB}} \quad z_D^{\mathrm{UB}}], \tag{3.1b}$$

where $\mathbf{z}^{\mathrm{LB}}$ and $\mathbf{z}^{\mathrm{UB}}$ are $1 \times D$ row vectors. Therefore, the $N \times D$ dimensional complete set of points, $\mathbf{Z}$ obeying the lower and upper bounds, $(\mathbf{z}^{\mathrm{UB}}, \mathbf{z}^{\mathrm{LB}})$, is obtained as:

$$\mathbf{Z} = (\mathbf{z}^{\mathrm{UB}} - \mathbf{z}^{\mathrm{LB}}) \odot \mathbf{R} + \mathbf{z}^{\mathbf{LB}}, \tag{3.2}$$

where $\odot$ indicates Hadamard product (element-wise multiplication). The row cardinality of the set, $|\mathbf{Z}|$, is equal to N and the column cardinality of the set is equal to D. The complete set can be also written as matrix form as:

$$\mathbf{Z} = \begin{bmatrix} z_1^1 & \cdots & z_D^1 \\ \vdots & \ddots & \vdots \\ z_1^N & \cdots & z_D^N \end{bmatrix}. \tag{3.3}$$

The set $\mathbf{Z}$ is a $N \times D$ matrix with N samples (rows) in D dimensions (columns). The set $\mathbf{Z}$ is then divided into feasible and infeasible sets. All the inequality constraints are

evaluated at the sample points included in set **Z**. Then, we can assign the points to the feasible set, **X**, if all the constraints are satisfied, $\mathbf{g}(\mathbf{X}) \leq \mathbf{0}$, and to the infeasible set, **Y**, if any one of the constraints is violated, $\mathbf{g}(\mathbf{Y}) > \mathbf{0}$.

At this point, readers may ask why and how the constraints are used to assign sample points as feasible and infeasible if the goal of this work is to determine the unknown constraints. To provide clarification on this issue, it should be noted that we need known true constraints to determine whether the constraints are identified correctly or not by our constraint-identification method. At this stage, by comparing the true and identified constraints, we will prove that our algorithm works properly with various constraint-identification examples (Chapter 6). In further examples, we may employ data sets (sample points) obtained via plant simulation directly without knowing / needing the explicit mathematical form of the constraints, by taking the converged simulation results as feasible points and unconverged simulation results as infeasible points, thus, mimicking the data collection from an actual plant and task of figuring out its feasible-region boundaries, experimentally as in real life. Thus, at this point, the readers should assume that the feasible and infeasible points (regions) come from unknown constraints even though we generate samples from known constraints. This is the only way to prove that our constraint-identification algorithm works as intended. Also, it should be noted that, the constraint-identification algorithm is totally blind to constraints themselves forming the region, the constraint-identification algorithm only sees the sample points in the regions and their labels as feasible and infeasible. We can designate the set of feasible and infeasible points in a more concrete way:

$$\mathbf{X} = \left\{\mathbf{x}^l \middle|\, \mathbf{x}^l \subset \mathbf{Z} \text{ and } \mathbf{g}(\mathbf{x}^l) \leq \mathbf{0};\ l = 1, \dots, \mathrm{N}^{\mathrm{f}}\right\}, \tag{3.4a}$$

$$\mathbf{Y} = \left\{\mathbf{y}^k \middle|\, \mathbf{y}^k \subset \mathbf{Z} \text{ and } \mathbf{g}(\mathbf{y}^k) > \mathbf{0};\ k = 1, \dots, \mathrm{N}^{\mathrm{i}}\right\}, \tag{3.4b}$$

where N^{f} and N^{i} are the number of feasible and infeasible points, respectively. $\mathbf{x}^l$ and $\mathbf{y}^k$ are $1 \times \mathrm{D}$ row vectors containing the l^{th} and k^{th} feasible and infeasible sample points in each coordinate direction, respectively. More precisely, the row cardinality of the feasible set, $|\mathbf{X}|$, is N^{f} while the row cardinality of the infeasible set, $|\mathbf{Y}|$, is N^{i}. The column cardinalities of the sets **X** and **Y** are both D. At this point it can be also beneficial to remind that all points in the feasible set, **X**, fall into the joint feasible region of the inequality constraints, and all points in the infeasible set, **Y**, fall into the joint infeasible region of the

inequality constraints, as thoroughly discussed in Chapter 2. The sets of feasible and infeasible points can be represented in their matrix forms as:

$$\mathbf{X} = \begin{bmatrix} x_1^1 & \cdots & x_D^1 \\ \vdots & \ddots & \vdots \\ x_1^{N^f} & \cdots & x_D^{N^f} \end{bmatrix}, \tag{3.5a}$$

$$\mathbf{Y} = \begin{bmatrix} y_1^1 & \cdots & y_D^1 \\ \vdots & \ddots & \vdots \\ y_1^{N^i} & \cdots & y_D^{N^i} \end{bmatrix}, \tag{3.5b}$$

where $\mathbf{X}$ is the $N^f \times D$ matrix and $\mathbf{Y}$ is the $N^i \times D$ matrix, both of which are in the D dimensional space. Each row of these matrices corresponds to sample points in D coordinates and each column of these matrices corresponds to sample points in the coordinate direction corresponding to that column.

Lastly, in addition to these notations, it can be also deduced that since $\mathbf{Z} = \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix}$ the cardinality of the complete set corresponds to $N = N^f + N^i$. The following set of ratios gives the fraction of the feasible and infeasible points:

$$F^f = N^f/N, \tag{3.6a}$$

$$F^i = N^i/N. \tag{3.6b}$$

After introducing these notations, we can return to detailed MCS analyses to examine the advantages and disadvantages of RS, SLHS, and MS methods with regard to constraint identification. It should be also noted that with SLHS, the number of points generated may be slightly greater than the specified N since the number of stratum in each dimension, $N^{(1/D)}$, must be an integer number. However, depending on N and D values, $N^{(1/D)}$ may be non-integer and the total number of samples generated becomes slightly greater than N when $N^{(1/D)}$ is rounded up to nearest integer. For example, for $D = 2$ and $N = 500$, the number of stratum in each dimension becomes $N^{(1/D)} = 22.361$. However, this value becomes 23 when rounded, and thus, the total number of samples generated, N, becomes 529. The same is true for MS as well, since the number of mesh points in each dimension is selected as $N^{(1/D)}$ and this number is rounded up in mesh generation as in SLHS. However, in order to preserve the uniformity of the presentations, only the targeted value of the total number of the samples, N, will be reported for SLHS and MS.

Figure 3.1 delineates RS of feasible and infeasible regions with the various cardinalities of the set of sample points. At first stage, we examine how the number of samples in the set has an effect on the constraint-identification task (here, in this chapter, visual constraint recognition and visual region recognition). For example, with $N = 50$ samples and with the set of inequality constraints introduced in Chapter 2 (Equation (2.1)), the sampled points are distributed as shown in Figure 3.1a, which is an illustration of under-sampling. Clearly, with only 50 samples, the feasible region of inequality constraints is not very precise, i.e., the boundary of feasible region (the boundary between the feasible and infeasible point sets) is ambiguous. If the RS sampling is repeated with $N = 500$ and $N = 1000$, Figure 3.1b and Figure 3.1c are obtained. As can be seen, as the number of sample points increases, the feasible-region boundary becomes more crisp. With $N = 10000$ the samples are distributed as shown in Figure 3.1f, which is an illustration of over-sampling. In such a situation, it is more obvious that there are more than necessary points to make boundary zone crisp. As can be seen from Figure 3.1d and Figure 3.1e, $N = 2500$ and $N = 5000$ samples suffice to reveal the boundary of feasible region. Therefore, 10000 samples provide only unnecessary improvement for revealing the feasible-region boundary at the expense of extra computational load.

Although the feasible boundary is more obvious with increasing number of sample points, constraint-identification task will be formulated as optimization problem (Chapter 5) and thus the objective function must be evaluated over N points. Therefore, large N values yield high CPU times. To avoid this situation and to reveal the feasible-region boundary sufficiently close, the sample size should be selected judiciously. However, as can be seen from this set of figures, only the points around the boundary play more crucial role in detecting constraints. Thus, a scheme for the reduction of sample size will be proposed and discussed in the next section.

Furthermore, with $N = 50$ samples, it is difficult to observe the feasible-region boundary due to under-sampling as can be seen from Figure 3.1a. In such a situation, the assessment of the sampling method can misguide readers in terms of its effects on constraint-identification task. The same is true for over-sampling which is represented by Figure 3.1f as well. Looking at Figure 3.1b and Figure 3.1c, where the number of samples is relatively low, one can see that uneven randomness accommodated by simple RS may

cause inadequate capture of the boundary zone of the constraints since there are gaps between some cluster of points and these gaps may be in the vicinity of the boundary zone, thus making it difficult to trace the constraints' boundary closely. Due to such gaps and clustering of the sample points which is inherent in RS, the corners of the feasible region may not be closely filled with sample points in such a way that sample points are enough to reveal the pattern of the feasible region completely, especially at the sharp corners of the region. In other words, with $N = 500$ and $N = 1000$ the RS method may probably prevent the constraint-identification algorithm from capturing such ambiguous inequality constraints truly. From Figure 3.1d and Figure 3.1e, the cardinalities of the sample set seem to be enough for RS to reveal the feasible-region boundary. There is, however, no guarantee that it will completely reveal the feasible-region boundary each time when RS is repeated with $N = 2500$ or with $N = 5000$ due to randomness. Even though RS can considerably help the constraint-identification algorithm discover unknown inequality constraints correctly with $N = 2500$ and $N = 5000$, the computational load can be significantly reduced with other sampling methods, meaning that with the same cardinality of the set, i.e., $N = 1000$, other methods may allow the constraint-identification algorithm to better fathom the obscure inequality constraints.

The SLHS, developed for alleviating the shortcoming of simple RS method, is depicted using the same set of inequality constraints and the same total numbers of sample points. Figure 3.2 below delineates SLHS with various cardinalities of the sample set. In this situation, as can be observed from this set of figures, the effect of the cardinality of the set on revealing the feasible-region boundary is similar to set of figures in Figure 3.1, meaning that like RS, under-sampling is observed from Figure 3.2a while over-sampling is seen in Figure 3.2f and the more the number of sample points, the crispier is the feasible-region boundary. In comparison with RS method in terms of the arrangement of the sample points in the regions, SLSH locates sample points in a more regular way, minimizing the occurrence of empty spots, thereby representing the feasible-region boundary better with the same number of sample points employed with RS. Readers can easily observe this situation when making comparison between Figure 3.1b and Figure 3.2b or between Figure 3.1c and Figure 3.2c. If readers make comparison between Figure3.1b and Figure 3.2b, they can observe that the feasible-region boundary is not completely obvious in both but more crisp in the Figure 3.2b.

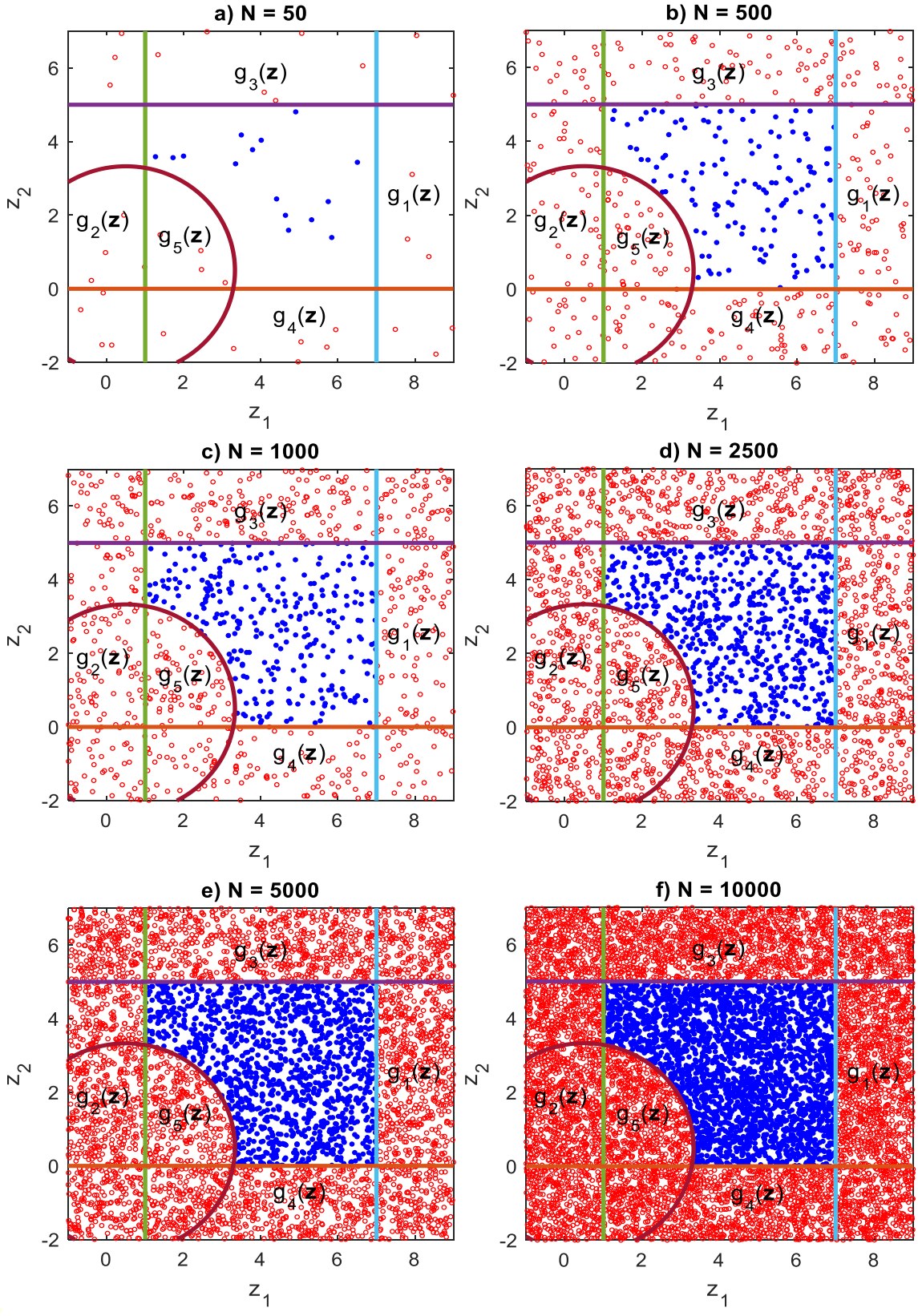

Figure 3.1. The illustration of Random Sampling.

Comparing Figure3.1c and Figure 3.2c, readers can observe that with SLHS the feasible-region boundary becomes more neatly apparent to the constraint-identification algorithm to capture unknown constraints with more ease, while with RS, the constraint-identification algorithm will probably fail to detect obscure inequality constraints. In addition, as can be observed from Figure 3.2d and Figure 3.2e, SLHS with $N = 2500$ and $N = 5000$ samples most likely allow the constraint-identification algorithm to decipher obscure constraints as well as RS, but with $N = 1000$ SLHS has more chance to reveal the feasible-region boundary than RS. Even though having this advantage, when the number of sample point is less than 2500, the SLHS may still fail to fill the sharp corners, which mainly unfold the type of inequality constraints joined together, i.e., linear-liner constraints junction or linear-circular constraints junction.

MS is depicted by the set of figures in Figure 3.3. When the cardinality of the sample points is considered, it is seen that the feasible region is not only gradually more obvious with increasing number of sample points likewise other methods, but also crispier with fewer number of sample points compared with other sampling methods. For instance, as can be seen from Figure 3.3b, the feasible-region boundary is more obvious than the ones in Figure 3.2b and Figure 3.1b even though the same number of samples is utilized. Although with RS and SLHS at $N = 500$, there is slim chance for the constraint-identification algorithm to detect unknown inequality constraints properly due to under-represented feasible-region boundary. With MS, at $N = 500$, the feasible-region boundary provides the constraint-identification algorithm with strong clue about what the inequality constraints are. Even though this method is the most basic sampling method, it seems to be better at revealing the pattern of the feasible-region boundary than other methods even with fewer sample points, especially at the sharp corners, since it does not accommodate randomness but the deployment of cartesian coordinate system. When considering MS, it should be noted that Figure 3.3a is still the illustration of under-sampling, and Figure 3.3f is still the illustration of over-sampling.

As a result of the comparisons of the sampling methods it can be deduced that in the context of constraint identification, MS method is superior to other methods without considering sample size reduction via boundary formation, which will be discussed in the next section.

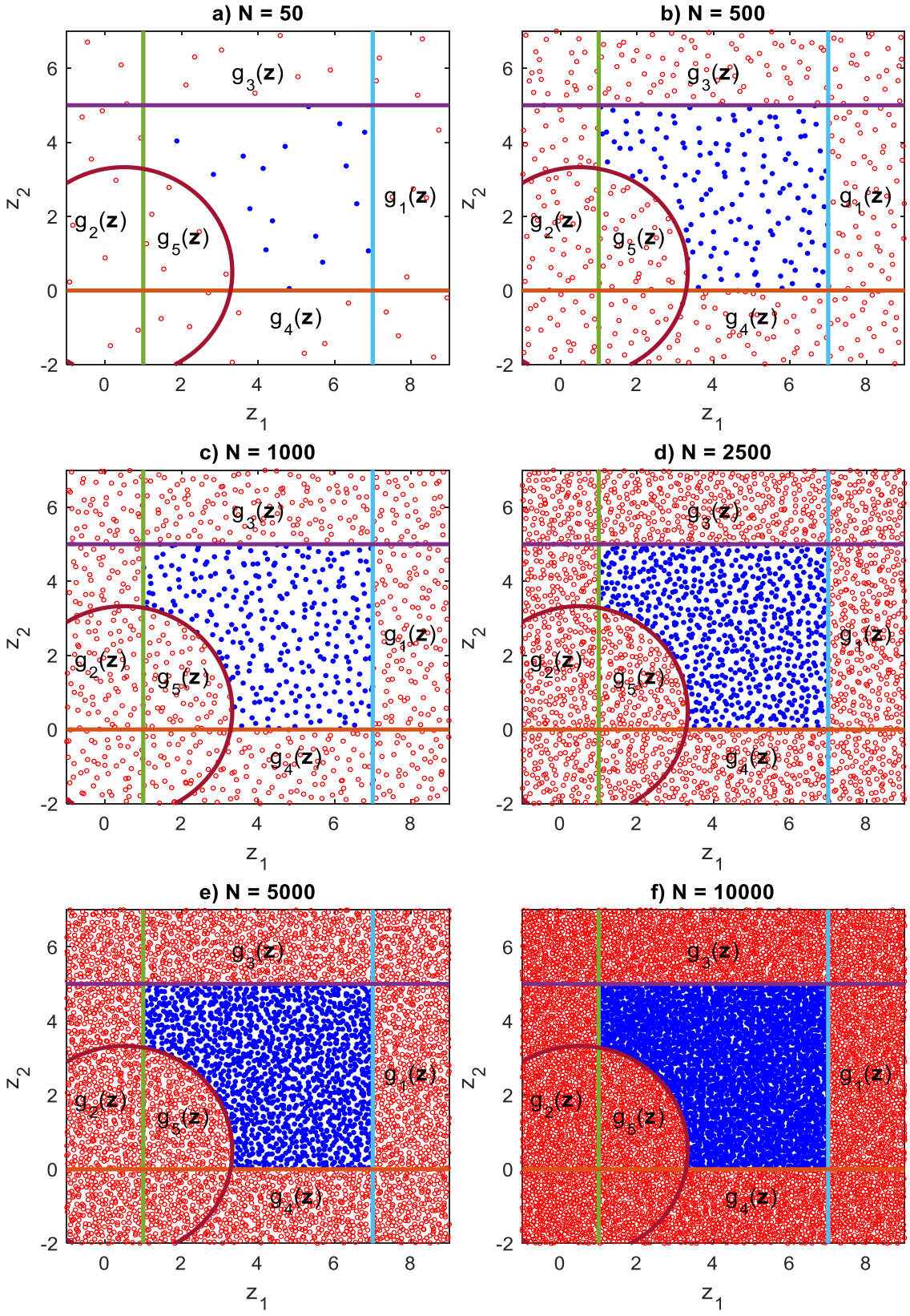

Figure 3.2. The illustration of Stratified Latin Hypercube Sampling.

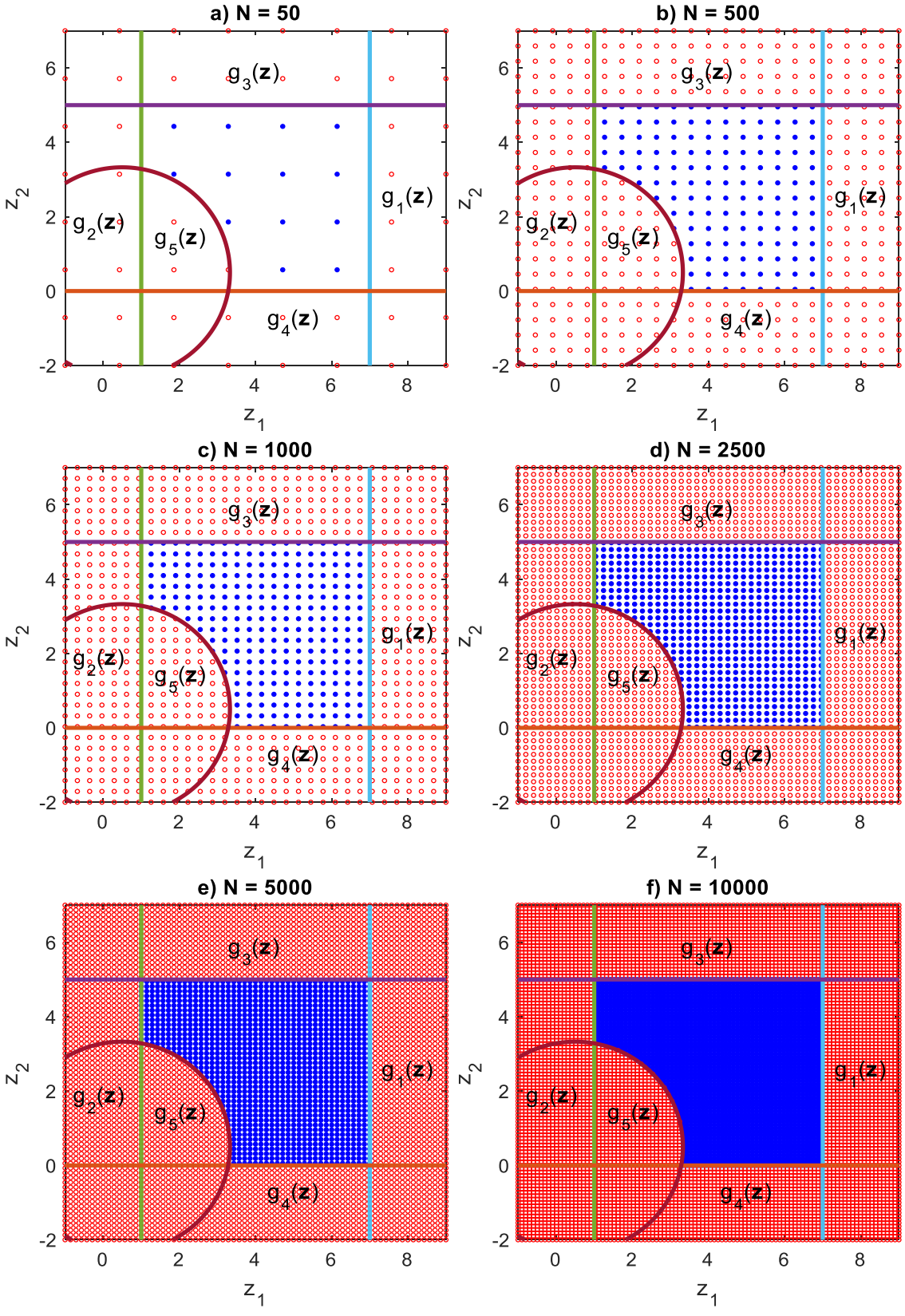

Figure 3.3. The illustration of Meshwise Sampling.

3.2. Sample-Size Reduction via Boundary-Zone Formation

Sample-size reduction is a key issue to expedite the constraint-identification process as mentioned in the previous section. As can be concluded from the previous section, MS proves more successful results in revealing the region with fewer sample points compared to other methods. Additionally, another ramification which can also be induced from the previous section is that the sample points around the feasible-region boundary play a more crucial role in revealing the shape of the union of the inequality constraints. This is because what demarcates the feasible-region boundary is actually the sample points in the vicinity of the boundary, which are low in number, rather than the other points deep within the feasible or infeasible regions, which are high in number. Relying on this observation, we can reduce the actual sample size used in constraint-identification algorithm without causing any negative impact on the representation of the boundary zone by keeping points only nearby the boundary region and removing the other points. Thus, the focus of this section is on how the sample points near the boundary zone can be extracted from the set of all feasible and infeasible points.

Sample-size reduction via boundary-zone formation includes the following steps. Firstly, a pre-specified p^{th} percentile (e.g., $p = 10\%$) of the minimum of the row cardinalities, $\min\{|\mathbf{X}|, |\mathbf{Y}|\} = \min\{N^f, N^i\}$, of the feasible ($\mathbf{X}$) and infeasible ($\mathbf{Y}$) sets is computed as $N^{dist} = p \times \min\{N^f, N^i\}$, which is rounded to the nearest integer. Next, the distances between each pair of feasible ($\mathbf{X}$) and infeasible ($\mathbf{Y}$) points are calculated using the Euclidean-distance metric as given by:

$$\mathbf{D_{XY}} = \text{dist}\{\mathbf{X}, \mathbf{Y}\}. \tag{3.7}$$

This gives us the $N^f \times N^i$ distance matrix, $\mathbf{D_{XY}}$, of the feasible points to infeasible ones or vice versa.

Finally, the distances, $\mathbf{D_{XY}}$, between each pair of feasible ($\mathbf{X}$) and infeasible ($\mathbf{Y}$) points are sorted from smallest to largest and the $\mathbf{X}$ and $\mathbf{Y}$ pairs corresponding to the smallest N^{dist} (p^{th} percentile) of distances, $\mathbf{D_{XY}}$, are retained as the boundary points, $\mathbf{X}^b$ and $\mathbf{Y}^b$, and the others are eliminated. Since the selected p^{th} percentile is much less than 100% (e.g., 10%), only the closest p^{th} percentile of the pairs of $\mathbf{X}$ and $\mathbf{Y}$ are thus selected. This boundary-zone selection process can be represented mathematically as:

$$\mathbf{X}^{\mathrm{b}} = \left\{\mathbf{X}, \mathbf{Y} \mid \ |\mathbb{S}\{\mathbf{D_{XY}}\}| = \mathrm{N}^{\mathrm{dist}}\right\}, \tag{3.8a}$$

$$\mathbf{Y}^{\mathrm{b}} = \left\{\mathbf{Y}, \mathbf{X} \mid \ |\mathbb{S}\{\mathbf{D_{XY}}\}| = \mathrm{N}^{\mathrm{dist}}\right\}, \tag{3.8b}$$

where the sorting operator $\mathbb{S}\{\mathbf{D_{XY}}\}$ should be read as "sort the row distances between **X** and **Y**, from lowest to highest". The term $|\mathbb{S}\{\mathbf{D_{XY}}\}| = \mathrm{N}^{\mathrm{dist}}$ implies the selection of smallest $\mathrm{N}^{\mathrm{dist}}$ distances and their corresponding unique elements from the sets **X** and **Y** to form the re-indexed sets $\mathbf{X}^{\mathrm{b}}$ and $\mathbf{Y}^{\mathrm{b}}$. In this way, for instance, by decreasing the p^{th} percentile value, which yields lower $\mathrm{N}^{\mathrm{dist}}$ value, we can decrease the number of boundary points assigned to sets $\mathbf{X}^{\mathrm{b}}$ and $\mathbf{Y}^{\mathrm{b}}$, and thus select only the closest **X** and **Y** pairs, from the original entire sets **X** and **Y**. It should be remembered from Section 3.1 that the row cardinalities of **X** and **Y** may be different, N^{f} and N^{i}, and thus the use of a common p^{th} percentile may also yield different cardinalities for $\mathbf{X}^{\mathrm{b}}$ and $\mathbf{Y}^{\mathrm{b}}$.

Before examining sample-size reduction via boundary-zone formation deploying SLHS and MS with different values of percentile and the cardinalities of the set of all points, it should be also noted that the cardinality of the boundary set of all points and the cardinalities of the boundary set of feasible and infeasible points are represented by N^{b}, N^{fb}, N^{ib}, respectively. For the sake of clarity, it should be also highlighted that initial sample size will refer to the cardinality of the set of all points, i.e., N, and reduced sample size will refer to the cardinality of the boundary set of all points, i.e., N^{b}.

SLHS and MS were executed with $\mathrm{N} = 50$, $\mathrm{N} = 500$, $\mathrm{N} = 1000$, $\mathrm{N} = 2500$, $\mathrm{N} = 5000$ and $\mathrm{N} = 10000$ in the previous section without sample-size reduction as can be seen in Figure 3.2 and Figure 3.3 respectively. In this section, sample-size reduction via boundary formation will be applied to these regions depicted in Figure 3.2 and Figure 3.3 except under-sampled and over-sampled ones. Under-sampling and over-sampling were thoroughly discussed in the previous section and it was concluded that the former remained insufficent to completely represent the region, while the latter also provided more points than necessary to reveal the boundary zone. Thus, these situations will be discarded in this section.

Figure 3.4 below illustrates applications of sample-size reduction via boundary-zone formation to the regions originally formed by SLHS with $\mathrm{N} = 500$, $\mathrm{N} = 1000$, $\mathrm{N} = 2500$

and $N = 5000$. For all cases in Figure 3.4, the value of percentile, i.e., p, which adjusts the reduced sample size is set to one percent to investigate only the effects of the number of samples on boundary-zone formation. The reduced sample sizes, N^b, become 93, 167, 429, 856 for the initial sample sizes, N, of 500, 1000, 2500 and 5000 as can be seen in Figures 3.4a-d, respectively. Initial sample sizes are considerably reduced without causing any deterioration at the previously fully-sampled boundary region and reveals the union of the inequality constraints very well. Thus, deployment of these reduced zones for constraint identification allows the optimizer to reach an optimal solution faster. These numbers also imply that the larger the initial sample size, the more apparent the extracted boundary zone with the fixed value of the percentile. Therefore, initial sample size should be determined suitably for the sample-size reduction technique to extract the most apparent boundary zone. In other words, sample-size reduction does not make any positive contribution to the representation of the boundary zone if the initial sampling is insufficient. For instance, as can be seen in Figure 3.4a, points around boundary zone remain insufficient to truly represent boundary zone due to small initial sample size. As can be seen in Figure 3.4d, the points near the boundary zone describe the actual boundary zone more correctly because of large initial sample size compared to other sub-figures in Figure 3.4. More precisely, the initial sample size should be specified without any consideration about the computational load caused by large initial sample size since the proposed reduction method will achive considerable reduction of the sample size without any negative effect on the revealed contour of the union of the inequality constraints, if the value of percentile is suitably selected.

The effect of the selection of the value of percentile is depicted in Figure 3.5. For all the cases depicted in the figure, initial sample size, N, is set to 5000 which does not lead to excessive computational load but also avoids initial under-representation of the boundary zone. In Figure 3.5 below, sample-size reduction with $p = 0.5$, $p = 1$, $p = 5$ and $p = 10$ is applied to the regions initially formed by SLHS. As can be seen in Figure 3.5a, the thinnest feasible boundary zone is extracted from the entire set with $p = 0.5$, whereas the thickest boundary zone is obtained with $p = 10$ as shown in Figure 3.5d. The sample size is considerably reduced from $N = 5000$ to $N^b = 602$ with $p = 0.5$. In addition, sample size is also reduced from $N = 5000$ to $N^b = 2585$ with $p = 10$. In the other situations

which can be seen in Figure 3.5b and Figure 3.5c, reduced sample sizes, N^b, become 856 and 1885, respectively.

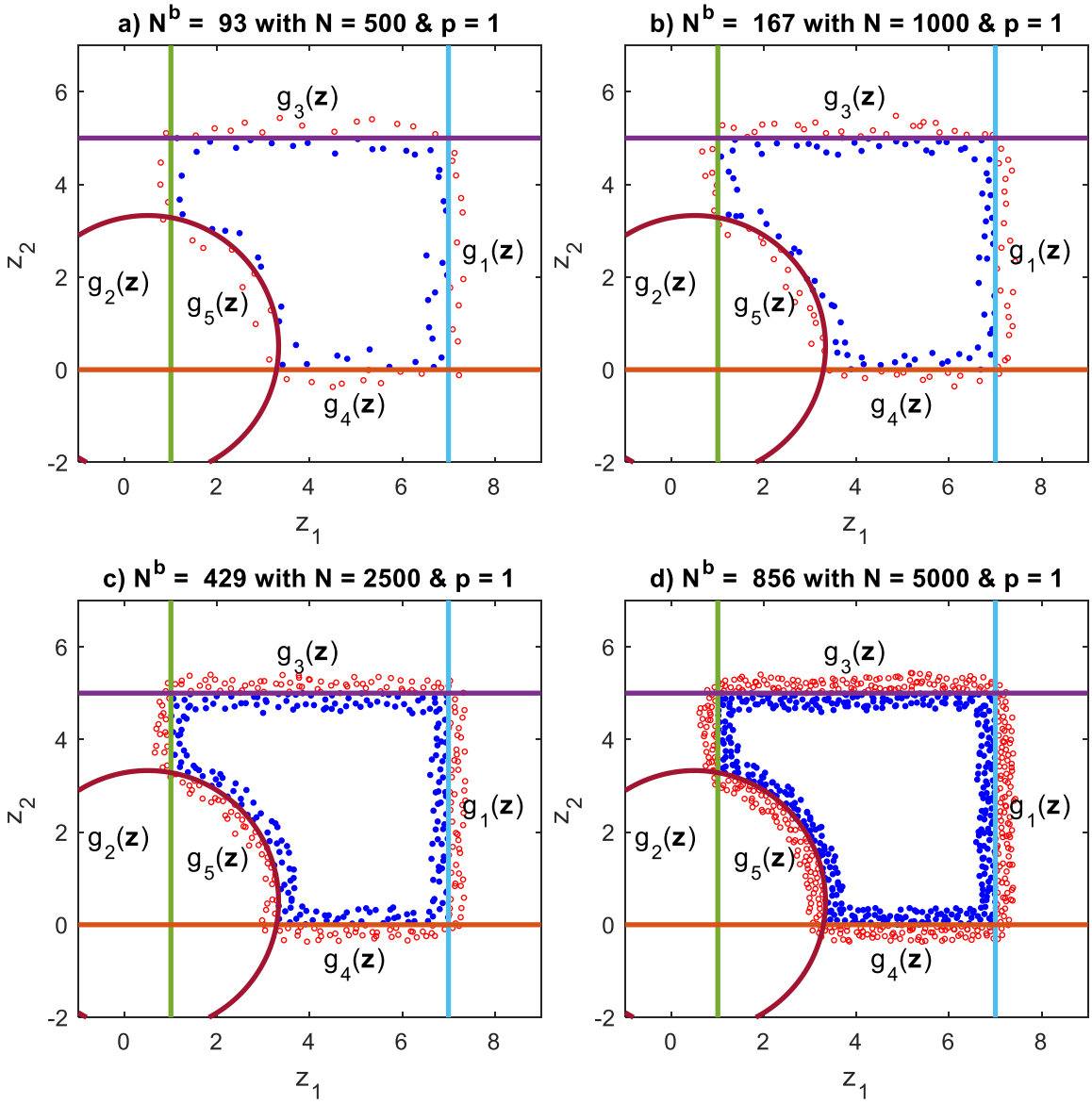

Figure 3.4. Effect of initial sample size on boundary formation under constant percentile value for Stratified Latin Hypercube Sampling.

The reduced sample size decreases with decreasing value of percentile as explained before. In other words, thicker boundary zone is obtained, if the initial sample size is fixed, as the value of the percentile increases. In addition, these reductions in sample sizes result in significantly low CPU time in constraint identification. Morever, there is also no difference between sub-figures depicted by Figure 3.5 in terms of represantation of the union of the

inequality constraints. As discussed earlier, describing boundary region correctly depends on sampling method used and initial sample size specified, and the sample-size reduction with $p = 0.5$, $p = 1$, $p = 5$ and $p = 10$ does not also cause any negative impact on the demonstration of the boundary zone. Although there is no deterioration at the representation of the boundary zones in Figure 3.5, choosing the value of the percentile too small leads to losing too much information. Thus, it can be better to adjust the value of the percentile above the one demonstrated in Figure 3.5a, i.e., $p > 0.5$, when the region is sampled using SLHS.

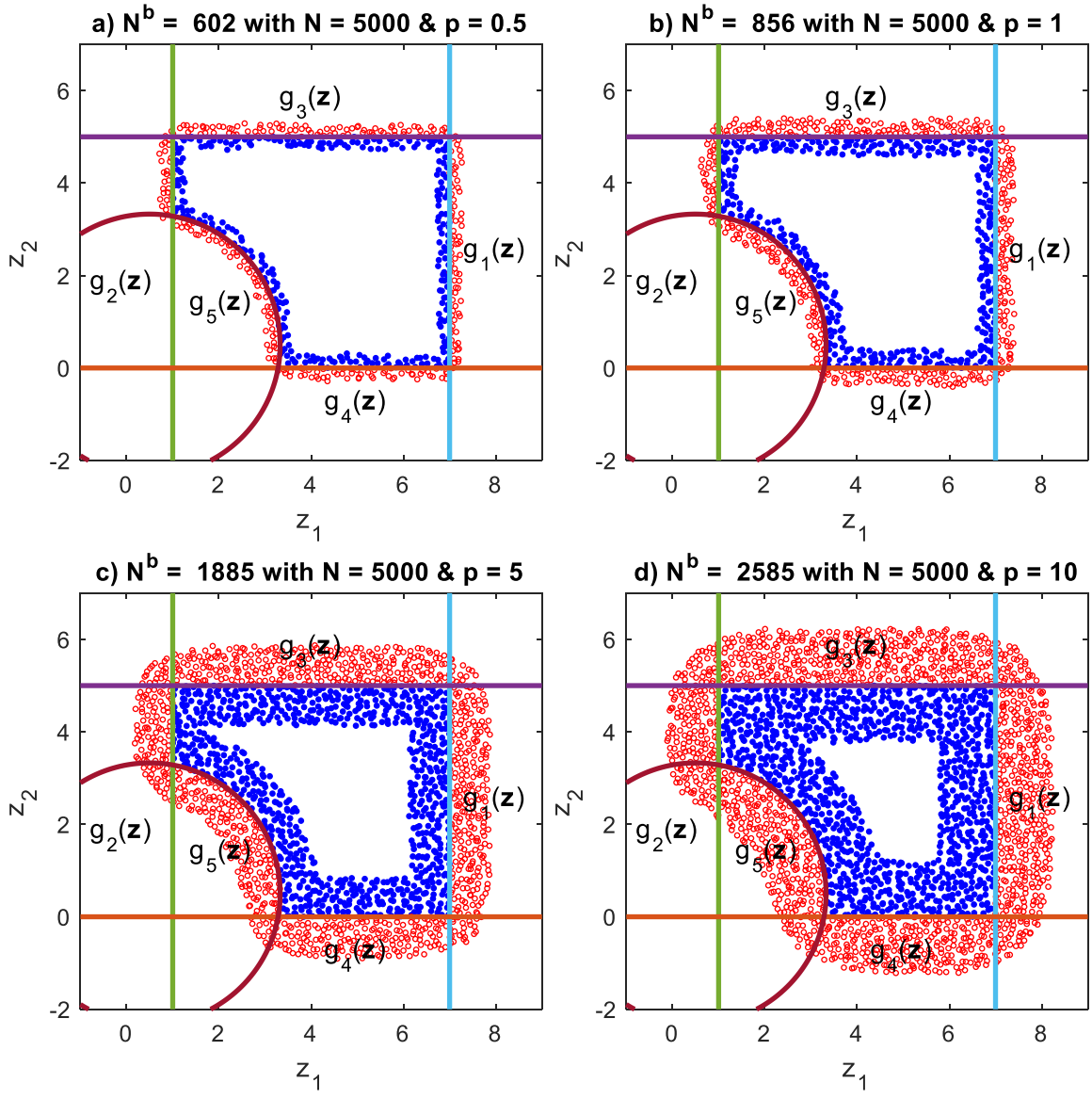

Figure 3.5. Effect of percentile value on boundary formation under constant initial sample size for Stratified Latin Hypercube Sampling.

Figure 3.6 below demonstrates the boundary-zone formation with $p = 1$ to the regions originally formed by MS with $N = 500$, $N = 1000$, $N = 2500$ and $N = 5000$. The only difference between the regions shown in Figure 3.6 and the ones in Figure 3.4 is the utilization of MS instead of SLHS. Likewise, the first and last sub-figures in Figure 3.4, the most subtle boundary zone caused by small initial sample size emerges in Figure 3.6a, while the most visible boundary zone induced by large initial sample size arises in Figure 3.6d. Additionally, the reduced sample sizes , N^b, become 92, 139, 422 877 while the initial sample sizes, N, equal to 500, 1000, 2500 and 5000 as can be seen in Figure 3.6a-d, respectively. These reductions in the initial sample sizes also significantly reduce CPU time in constraint identification. Even though MS seems to be good at revealing true boundary zone with smaller initial sample size, initial sample size should be selected as large as possible to guarantee that boundary region is sampled sufficiently before boundary-zone formation.

Figure 3.7 demonstrates the boundary-zone formation with various percentile values for the regions initially formed by MS. The initial sample size, N, is also set to 5000 so as to make sure that initial sample size does not under-represent the feasible region. In Figure 3.7, sample-size reduction via boundary-zone formation with $p = 0.5$, $p = 1$, $p = 5$ and $p = 10$ is applied to the regions initially formed by MS. The same trend in the regions shown in Figure 3.5 can be easily observed from sub-figures in Figure 3.7, in terms of the thickness of the boudary zone.

The thinnest boundary zone is obtained with the smallest value of the percentile, i.e., $p = 0.5$, whereas the thickest boundary zone is achieved with the largest value of the percentile, i.e., $p = 10$, when the initial sample size is fixed, as can be seen in Figure 3.7a and Figure 3.7d respectively. The sample size, as can be seen in Figure 3.7a, is considerably reduced from $N = 5000$ to $N^b = 606$ with $p = 0.5$. In the other situation depicted in Figure 3.7d, sample size is reduced from $N = 5000$ to $N^b = 2602$ with $p = 10$. Considerable reductions in the sample size are also achieved for the situations shown in Figure3.7b and Figure 3.7c, which means that the reduced sample sizes, N^b, become 877, 1876 while the initial sample sizes, i.e., N, are 5000 for both situations.

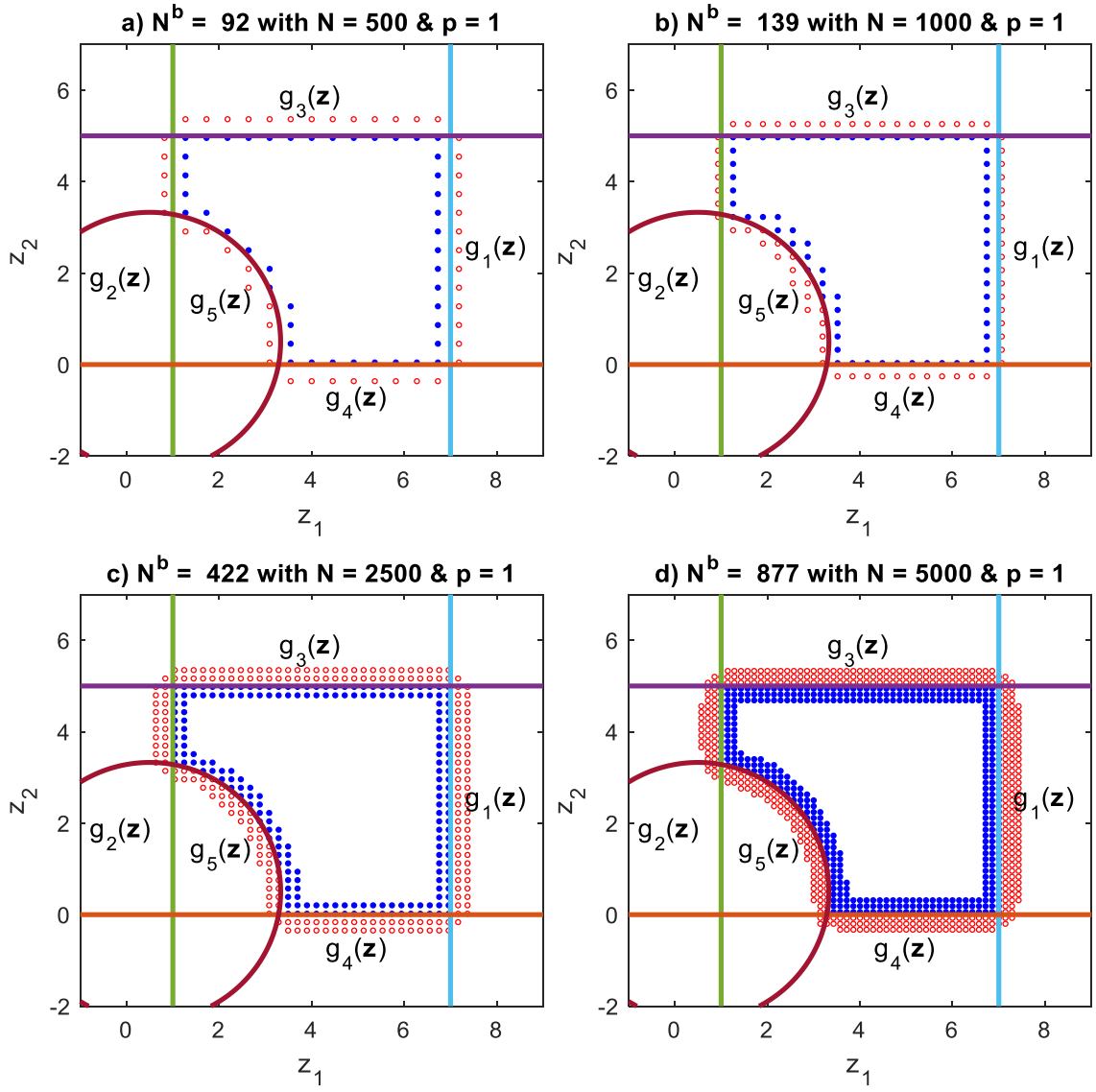

Figure 3.6. Effect of initial sample size on boundary formation under constant percentile value for Meshwise Sampling.

Such reduction in the sample size rather expedites the constraint-identification task. Sample-size reduction via boundary-zone formation with $p = 0.5$, $p = 1$, $p = 5$, $p = 10$ does not also damage the representation of the boundary zone initially sampled by MS with $N = 5000$. However, selecting the percentile too small like the one shown in Figure 3.7a, i.e, $p = 0.5$, causes too much information lost even though the most vital parts of the information, i.e., points around the boundary zone, are preserved. As also discussed earlier, the selection of the percentile value lower than, i.e., $p < 0.5$, may cause some boundary points to be removed, which, in turn, probably harms the success of the constraint-

identification algorithm. Thus, it can be better if the value of percentile is kept over the value shown in Figure 3.7a, i.e., $p > 0.5$, even when the region is sampled by MS.

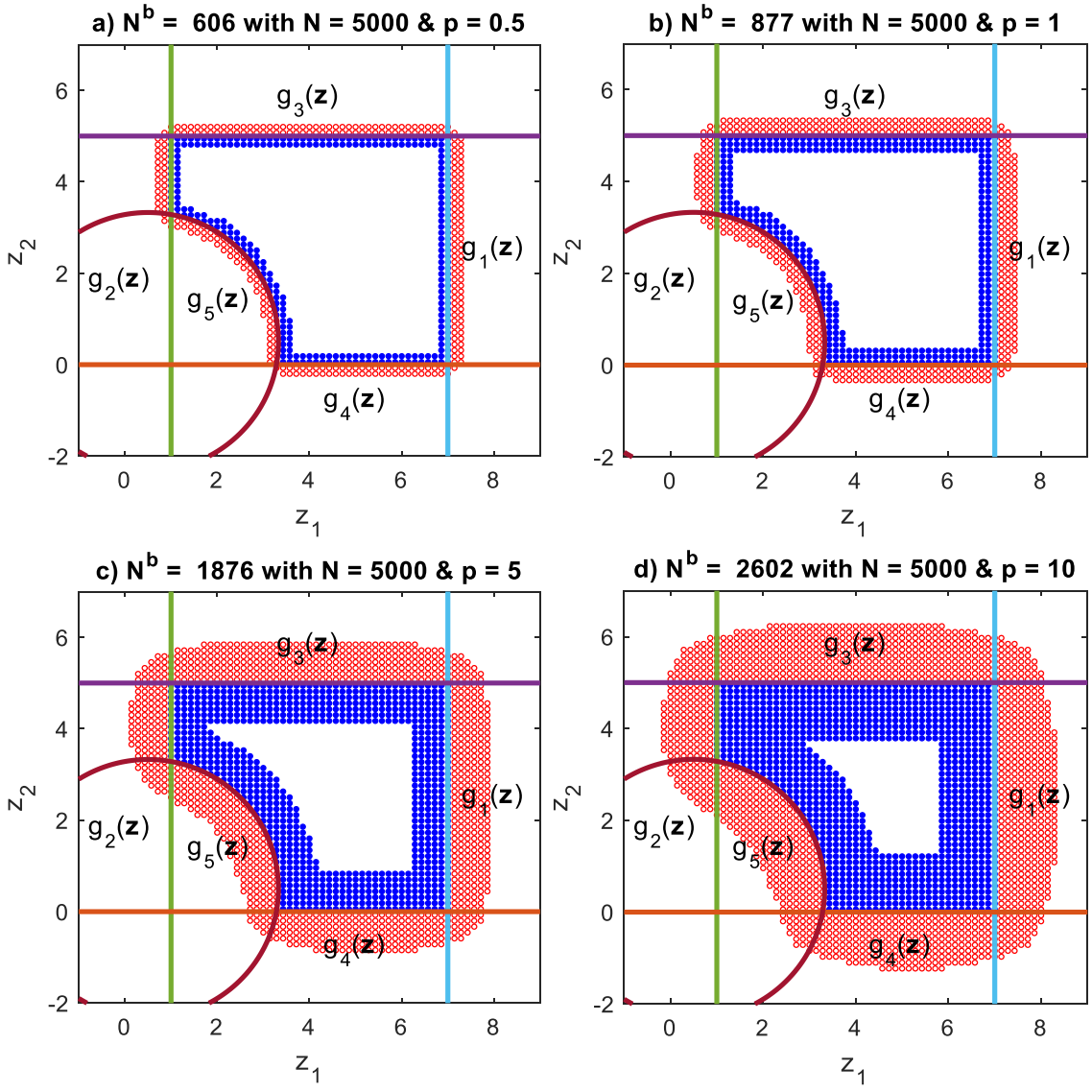

Figure 3.7. Effect of percentile value on boundary formation under constant initial sample size for Meshwise Sampling.

4. CONSTRAINT AGGREGATION AND THE KREISSELMEIER-STEINHAUSER FUNCTION

Constraint Aggregation (CA) is a method that allows one to embed several constraints, whose types can be equality or inequality, into one constraint with sacrificing only an acceptable (mostly negligible) amount of the feasible region from the true one. In other words, CA reduces the number of constraints, and thus, enables one to solve optimization problems with only one but highly-nonlinear constraint, possibly in a shorter time. This is because solving optimization problems including large number of constraints is computationally more expensive due to evaluation of constraints and their gradients. Even if the constraints of an original optimization problem are linear, which means that the problem is a Linear Programming (LP) problem if the objective function is also linear, the aggregated constraint obtained after applying CA technique to original constraints becomes a nonlinear constraint. Therefore, application of CA to an optimization problem can be at the expense of the change in the inherited characteristic of an optimization problem meaning that a LP problem must be solved as a Nonlinear Programming (NLP) problem since the aggregated constraint becomes highly nonlinear.

To apply CA to inequality constraints, $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$, there is no need to rewrite them. However, when it comes to equality constraints, $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, it is necessary to reformulate them as two inequality constraints with a small error margin to avoid the equality-constraint violation. More precisely, equality constraints are reformulated as given below:

$$+\mathbf{h}(\mathbf{x}) \leq \varepsilon, \tag{4.1a}$$

$$-\mathbf{h}(\mathbf{x}) \leq \varepsilon, \tag{4.1b}$$

where ϵ is a quite small error margin such as 10^{-2}. There are several types of the CA method among which the Kreisselmeier-Steinhauser (KS) function (Kreisselmeier and Steinhauser, 1980) is commonly employed with the purpose of lumping constraints into a single inequality constraint. Its functional form can be written as:

$$C_{KS}(g, \rho) = \frac{1}{\rho} \ln \left[\frac{1}{\alpha} \int e^{\rho g} d\Omega \right], \tag{4.2}$$

where α is a normalization parameter, which should be greater than zero. The parameter α is taken as one or $|\Omega|$ (Kennedy and Hicken, 2015).

When considered for finite sets of constraints, the KS functional is transformed into function form by replacing integral operator with summation. Thus, the KS function, which is the discrete version of the KS functional, is obtained. The KS function is found in two different forms in the literature, one of them underestimates (KS_U) the feasible region while the other overestimates (KS_O) the feasible region. These functions are, for m constraints of type $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$, respectively:

$$KS_U(\mathbf{x}, \rho) = \frac{1}{\rho} \ln\left(\sum_{i=1}^{m} e^{\rho g_i(\mathbf{x})}\right), \tag{4.3}$$

$$KS_O(\mathbf{x}, \rho) = \frac{1}{\rho} \ln\left(\frac{1}{m}\sum_{i=1}^{m} e^{\rho g_i(\mathbf{x})}\right). \tag{4.4}$$

Here, ρ denotes a tightening parameter for the KS function, which can take any value greater than or equal to zero. To attain a tighter feasible region for both types of the KS functions, the value of ρ should be large. The larger the value of ρ, the tighter the feasible region.

However, there are some pitfalls pertaining to increasing the value of ρ. The most crucial of them is the "exponential overflow" issue which makes the exponential term uncomputable. In addition to this, an optimization solver can be more easily trapped in local extrema when the feasible region represented by the KS function is too tight.

Before applied to inequality constraints forming a two-dimensional feasible region to dissect its capability to represent the feasible region, the KS function can be deployed to aggregate different functions given below as an example:

$$F_1(x) = (x - 2)^2 + 1, \tag{4.5a}$$

$$F_2(x) = 2(x - 1), \tag{4.5b}$$

$$F_3(x) = 2.5 \sin 3x - 0.5(x - 2)^2 + 3. \tag{4.5c}$$

Figure 4.1 demonstrates the aggregation of these functions using the underestimator form of the KS (Equation (4.3)) with two different ρ values.

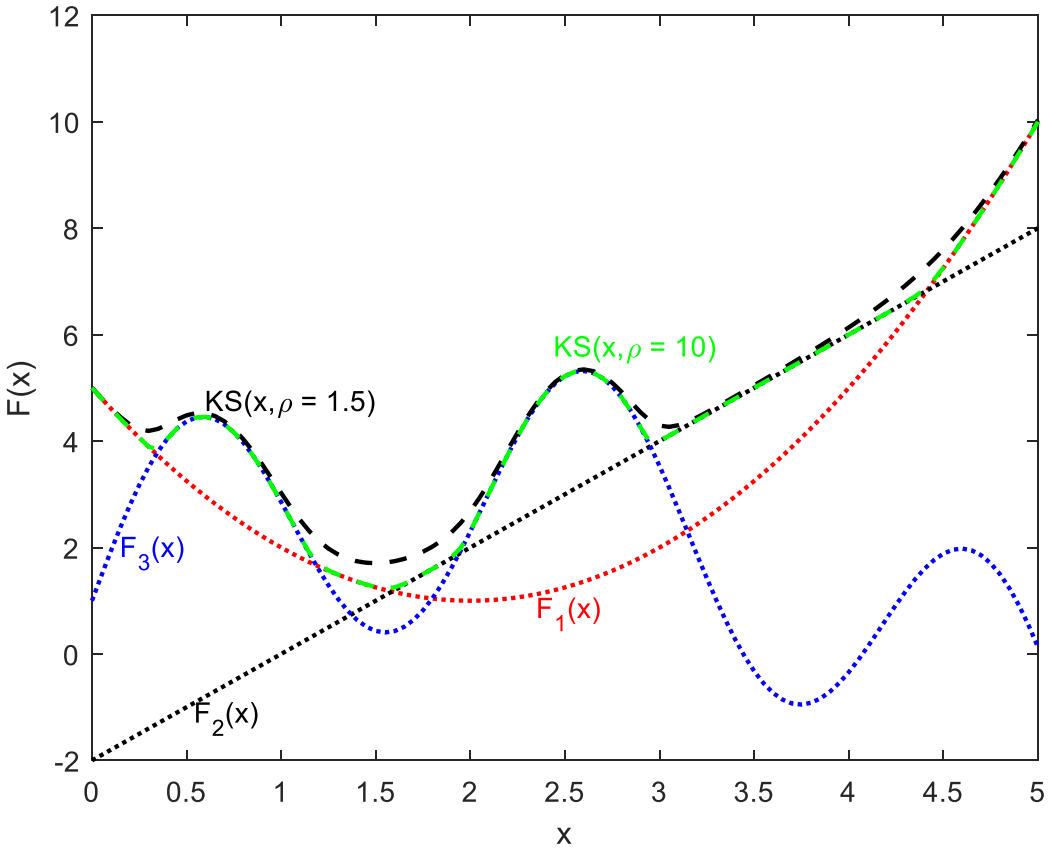

Figure 4.1. Application of the KS_U function to aggregate three functions.

To resolve exponential overflow problem, different types of the KS function modified from the original ones are introduced (Bloss et al., 1999):

$$\overline{KS}_O(\mathbf{x}, \rho) = g_{max} + \frac{1}{\rho}\ln\left(\frac{1}{m}\sum_{i=1}^{m} e^{\rho(g_i(\mathbf{x})-g_{max})}\right), \tag{4.6}$$

$$\overline{KS}_U(\mathbf{x}, \rho) = g_{max} + \frac{1}{\rho}\ln\left(\sum_{i=1}^{m} e^{\rho(g_i(\mathbf{x})-g_{max})}\right), \tag{4.7}$$

where

$$g_{max} = \max_i g_i(\mathbf{x}). \tag{4.8}$$

This KS function is also called the "modified KS function". Figure 4.2 demonstrates the aggregation of three functions given before using the modified KS_U function which underestimates the feasible region with two different ρ values.

Some remarks about the modified KS function ($\overline{KS}_U$) may be given at this point. This function is an underestimator of the feasible region and is identical to the standard

KS_U function except it prevents the exponential overflow problem even with considerably large values of ρ.

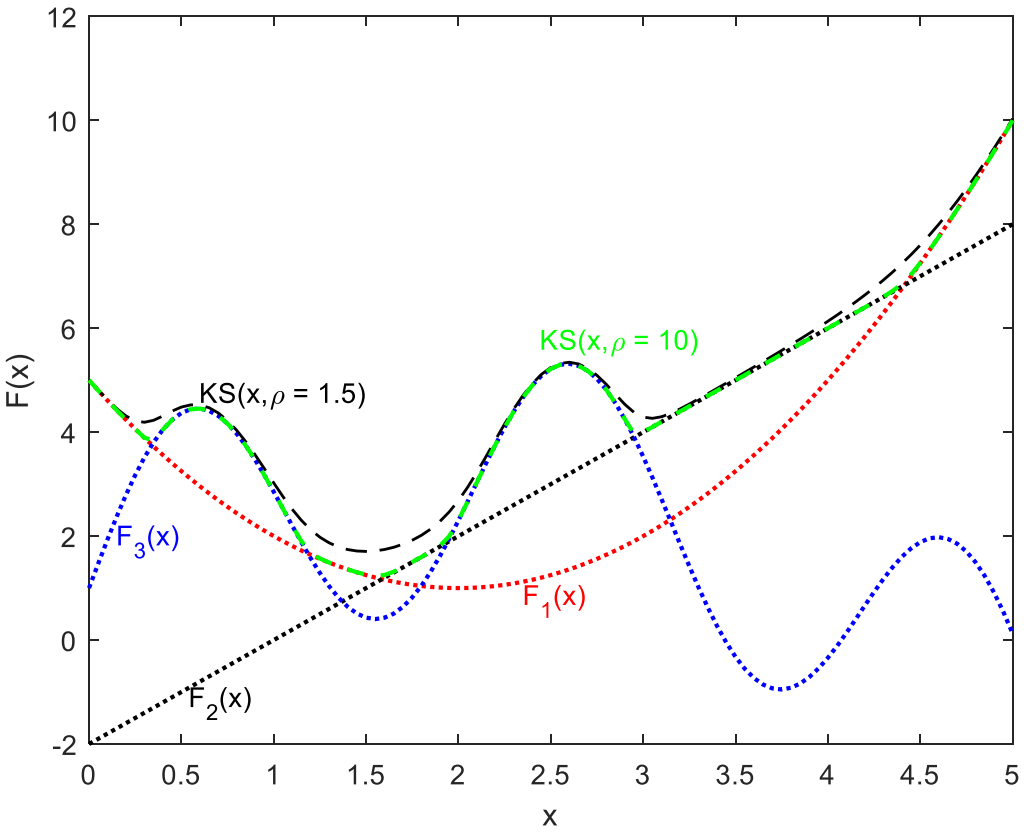

Figure 4.2. Application of the modified KS_U function to aggregate three functions.

The aggregated constraint becomes active, $\mathbb{C}(\mathbf{x}) = 0$, as all $g_i(\mathbf{x}) < 0$ become active, $g_i(\mathbf{x}) = 0$, individually. If any one of the original constraints gets violated, $g_i(\mathbf{x}) > 0$, it becomes the dominant term in the KS function summation due to large ρ value. The term $\exp(\rho g_i(\mathbf{x}))$ is roughly equal to zero for all inequality constraints less than zero, $g_i(\mathbf{x}) < 0$, due to large ρ. The modified KS function can be held well-behaved by assigning g_{max} as the upper bound to $g_i(\mathbf{x})$, so that exponential term, $\exp(\rho(g_i(\mathbf{x})-g_{max}))$, is less than or equal to one, and thus exponential-overflow issue is eliminated.

In cases where gradient-based optimization methods are utilized, the derivatives of the KS function may be necessary. The derivatives of both the original and modified KS functions with respect to **x** are given as follows (Raspanti et al., 2000), respectively:

$$\nabla_x KS = \frac{\sum_{i=1}^{m} e^{(\rho g_i)} \nabla_x g_i}{\sum_{i=1}^{m} e^{(\rho g_i)}}, \tag{4.9}$$

$$\nabla_x \overline{KS} = \nabla_x g_{max} + \frac{\sum_{i=1}^{m} e^{\rho(g_i - g_{max})} (\nabla_x g_i - \nabla_x g_{max})}{\sum_{i=1}^{m} e^{\rho(g_i - g_{max})}}. \tag{4.10}$$

After simplification, the derivative of the modified KS function becomes identical to the derivative of the original one because g_{max} is constant:

$$\nabla_x \overline{KS} = \frac{\sum_{i=1}^{m} e^{(\rho g_i)} \nabla_x g_i}{\sum_{i=1}^{m} e^{(\rho g_i)}}. \tag{4.11}$$

There also exist some additional properties of the KS function, which are represented as follow (Raspanti et al., 2000):

1. $KS(\mathbf{x}, \rho) \geq \max_i g_i(\mathbf{x})$, $\rho > 0$
2. $\lim_{\rho \to \infty} KS(\mathbf{x}, \rho) = \max_i g_i(\mathbf{x})$
3. $KS(\mathbf{x}, \rho_2) \geq KS(\mathbf{x}, \rho_1)$, $\forall x$ such that $\rho_1 > \rho_2 > 0$
4. The definitions stated by Equation (4.3) and Equation (4.4) are identical for any given value of g_{max}.
5. The gradient of the KS function with respect to **x** is independent of g_{max}.
6. For a convex region represented via a set of convex inequality constraints, $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$, the region with the definition $S(\mathbf{x}, \rho) \leq C_1$ is a convex region for any $\rho > 0$ and C_1 as well.
7. The original KS function tends to be insensitive to ρ while ρ is becoming very large.

These properties of the KS function have been proven in Raspanti et al. (2000). After discussing in detail, the behavior of both types of the KS function can be illustrated using an example set of inequality constraints. These example constraints forming a two-dimensional feasible region are:

$$g_1: \quad -x_1 \leq 0, \tag{4.12a}$$

$$g_2: \quad -x_2 \leq 0, \tag{4.12b}$$

$$g_3: \quad +x_1 - 9 \leq 0, \tag{4.12c}$$

$$g_4: \quad +x_2 - 6 \leq 0, \tag{4.12d}$$

$$g_5: \quad -0.3x_1 - 0.4x_2 + 2 \leq 0, \tag{4.12e}$$

$$g_6: \quad -0.4x_1 - 0.2x_2 + 1.5 \leq 0. \tag{4.12f}$$

It is possible to see the effect of ρ on representing the feasible region from both Figure 4.3 and Figure 4.4. Tighter feasible region is achieved by increasing the value of ρ. With decreasing value of ρ, overestimator form of the KS function surrounds the feasible region more and more towards outside and the underestimator form of the KS function more and more towards inside. In other words, with increasing value of ρ, overestimator form of the KS function surrounds the feasible region more accurately from outside and the underestimator form of the KS function more accurately from inside of the feasible region.

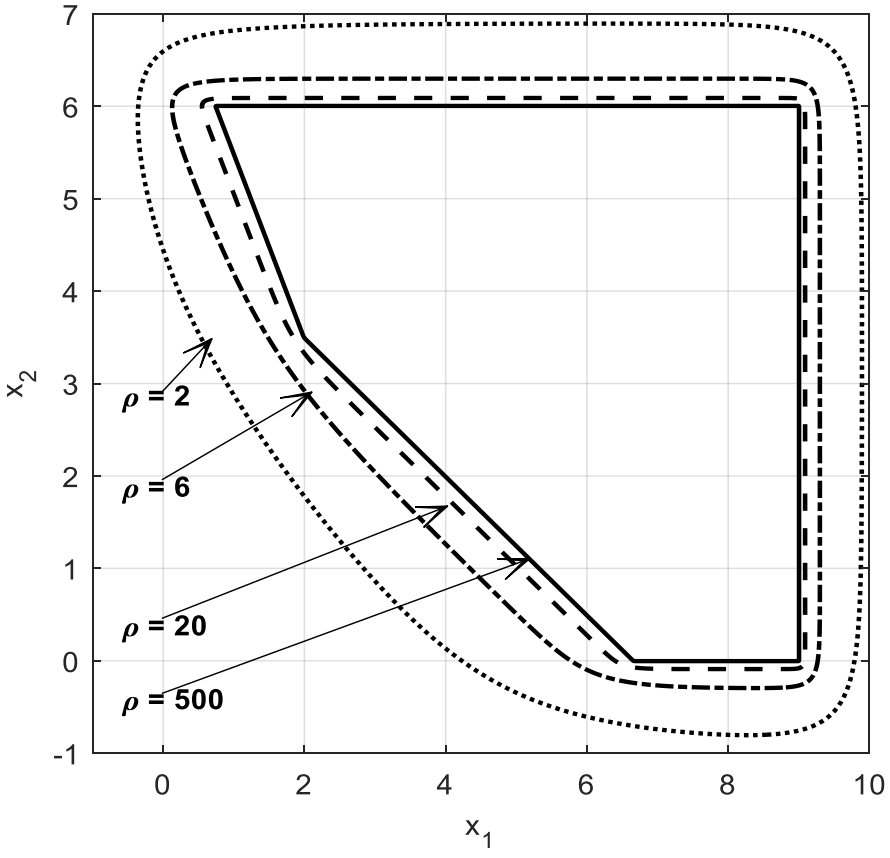

Figure 4.3. Application of overestimator form of the original KS function to the feasible region with different ρ values.

Lambe et al. (2017) lists some aggregation functions other than the KS function. However, these alternative aggregation functions are not widely used in the literature.

In this thesis, constraint identification task is formulated by taking advantage of the modified KS function underestimating the feasible region ($\overline{\mathrm{KS}}_{\mathrm{U}}$), and from now on it will be referred to simply as the "KS function". The use of a constraint aggregator in constraint identification is a must because the parameterization of constraints forming a feasible region and identification of the feasible and infeasible points by penalizing the objective function with respect to individual constraints are impossible. The only way to carry out the constraint identification procedure is to utilize an aggregator that combines all constraints in the form of $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ into a single constraint $\mathbb{C}(\mathbf{x}) \leq 0$ and evaluate the points as feasible and infeasible with respect to this single constraint. This point will be elaborated further in Chapter 5 of the thesis.

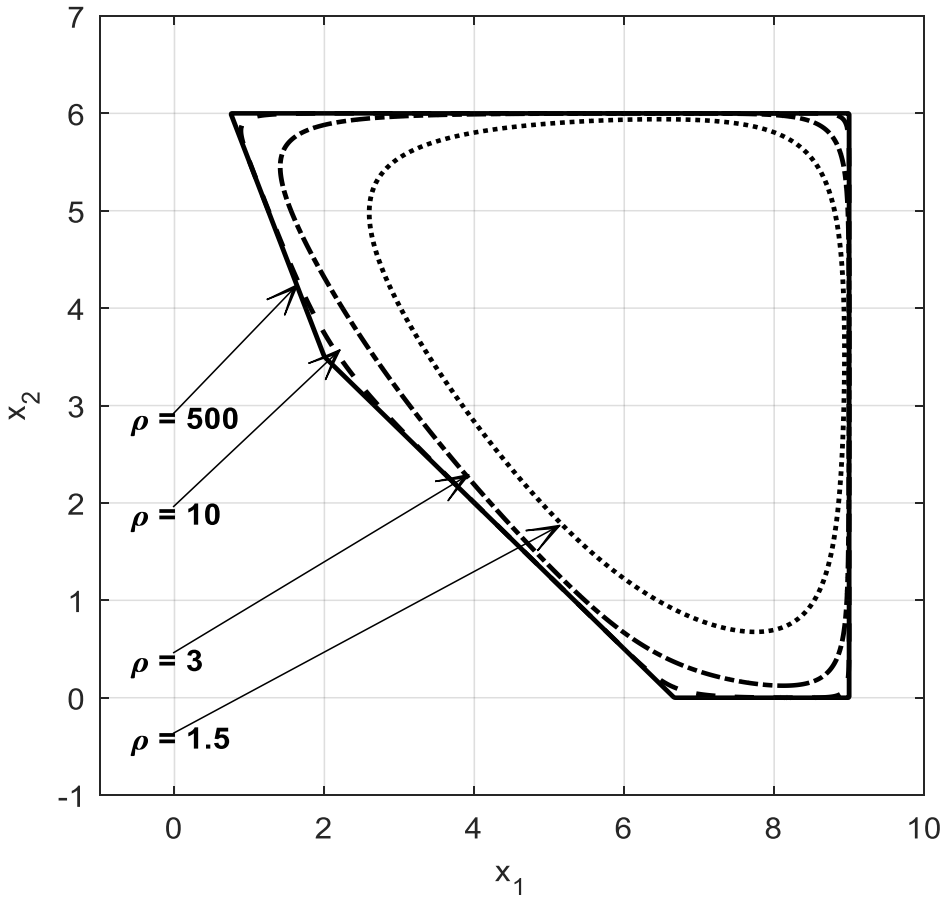

Figure 4.4. Application of underestimator form of the original KS function to the feasible region with different ρ values.

5. CONSTRAINT IDENTIFICATION

In this chapter, we will present the main topic of constraint identification, firstly by introducing parametric forms of some special constitutive inequality constraints such as bounds, linear, circular, and ellipsoidal ones. The formation of the objective function comes after the introduction to these parametric constitutive constraint forms. Lastly, in Section 5.3 we will briefly explain the optimization algorithm utilized in this thesis work.

5.1. The Constitutive Inequality Constraints

Before introducing the generic forms of certain type of constitutive inequality constraints, it can be beneficial to remind some notations about the set of feasible and infeasible points, which had already been thoroughly discussed in Chapter 3. The feasible set of points, $\mathbf{X}$, is the $\mathrm{N^f} \times \mathrm{D}$ matrix in D-dimensional space. Each of D-dimensional feasible points, $\mathbf{x}^l$ for $l = \{1, \dots, \mathrm{N^f}\}$, is the $1 \times \mathrm{D}$ row vector of matrix $\mathbf{X}$. Each dimension of feasible points, $\mathbf{x}_\mathrm{i}$ for $\mathrm{i} = \{1, \dots, \mathrm{D}\}$, is the $\mathrm{N^f} \times 1$ column vector of matrix $\mathbf{X}$. For the set of infeasible points, the infeasible set of points, $\mathbf{Y}$, is the $\mathrm{N^i} \times \mathrm{D}$ matrix in D-dimensional space. Each of D-dimensional infeasible point, $\mathbf{y}^k$ for $k = \{1, \dots, \mathrm{N^i}\}$, is the $1 \times \mathrm{D}$ row vector of matrix $\mathbf{Y}$. Each dimension of infeasible points, $\mathbf{y}_\mathrm{i}$ for $\mathrm{i} = \{1, \dots, \mathrm{D}\}$, is the $\mathrm{N^i} \times 1$ column vector of matrix $\mathbf{Y}$. These notations and dimensions for the set of feasible and infeasible points will be valid throughout the rest of this chapter.

Taking $\mathbf{Z}$ as either $\mathbf{X}$ or $\mathbf{Y}$, $\hat{\mathbf{g}}_\mathrm{j}(\mathbf{Z}, \boldsymbol{\theta}_\mathrm{j})$ must be read as "the $\mathrm{j^{th}}$ constitutive constraint evaluated at the $\mathrm{N^f}$ feasible points with its parameters $\boldsymbol{\theta}_\mathrm{j}$" when $\mathbf{Z}$ is $\mathbf{X}$, and as "the $\mathrm{j^{th}}$ constitutive constraint evaluated at the $\mathrm{N^i}$ infeasible points with the same parameters $\boldsymbol{\theta}_\mathrm{j}$" when $\mathbf{Z}$ is $\mathbf{Y}$. Thus $\hat{\mathbf{g}}_\mathrm{j}(\mathbf{X}, \boldsymbol{\theta}_\mathrm{j})$, which is the $\mathrm{N^f} \times 1$ column vector for each j, implies the values (outcomes) of the $\mathrm{j^{th}}$ constitutive inequality constraint evaluated at the feasible points, $\mathbf{X}$, with current values of parameters, $\boldsymbol{\theta}_\mathrm{j}$, and $\hat{\mathbf{g}}_\mathrm{j}(\mathbf{Y}, \boldsymbol{\theta}_\mathrm{j})$, which is the $\mathrm{N^i} \times 1$ column vector for each j, denotes the values (outcomes) of the $\mathrm{j^{th}}$ constitutive inequality constraints evaluated at the infeasible points, $\mathbf{Y}$, with the current values of parameters, $\boldsymbol{\theta}_\mathrm{j}$. These

definitions and dimensions for $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{j}})$ will also be valid whatever the specific type of inequality constraint is.

It should be noted that the feasible points, i.e., all rows of $\mathbf{X}$, satisfy all the individual constitutive inquality constraints, i.e., $\hat{\mathrm{g}}_{\mathrm{j}}(\mathbf{x}^{l}, \boldsymbol{\theta}_{\mathrm{j}}) \leq 0$ for $\forall\, l \in \{1, \dots, \mathrm{N}^{\mathrm{f}}\}$ and $\forall\, \mathrm{j} \in \{1, \dots, \mathrm{N}^{\mathrm{Type}}\}$. However, some infeasible points, i.e., some rows of $\mathbf{Y}$, may satisfy some - but not all- individual constitutive inequality constraints, i.e., $\hat{\mathrm{g}}_{\mathrm{j}}(\mathbf{y}^{k}, \boldsymbol{\theta}_{\mathrm{j}}) < 0$ for $\exists\, k \in \{1, \dots, \mathrm{N}^{\mathrm{i}}\}$ and $\exists\, \mathrm{j} \in \{1, \dots, \mathrm{N}^{\mathrm{Type}}\}$. Therefore, when $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Y}, \boldsymbol{\theta}_{\mathrm{j}}) > \mathbf{0}$ is written below, it implies that this is true at least for one row of $\mathbf{Y}$ but may not be true for all rows of $\mathbf{Y}$.

Firstly, the constitutive bound constraints, which are also called the "box constraints" or "side constraints", are given with respect to each feasible point, $\mathbf{x}^{l}$, (each row of $\mathbf{X}$) as: $\boldsymbol{\theta}^{\mathrm{L}} \leq \mathbf{x}^{l} \leq \boldsymbol{\theta}^{\mathrm{U}}$, and with respect to some infeasible points, $\mathbf{y}^{k}$, (some rows of $\mathbf{Y}$) as: $\mathbf{y}^{k} < \boldsymbol{\theta}^{\mathrm{L}}$ and $\mathbf{y}^{k} > \boldsymbol{\theta}^{\mathrm{U}}$ where the $1 \times \mathrm{D}$ vectors $\boldsymbol{\theta}^{\mathrm{L}}$ and $\boldsymbol{\theta}^{\mathrm{U}}$ are the lower and upper bounds on D coordinates. Each column of these vectors corresponds to each coordinate direction "i". The number of the constitutive box constraints, $\mathrm{N}^{\mathrm{Bnds}}$, when written in generic form, for feasible or infeasible points, depends on the column cardinality of the points (i.e., the problem dimension, D), meaning that we will utilize two bound constraints for each dimension, i.e., $\mathbf{x}^{l} \leq \boldsymbol{\theta}^{\mathrm{U}}$, and $\mathbf{x}^{l} \geq \boldsymbol{\theta}^{\mathrm{L}}$, for each feasible point, and $\mathbf{y}^{k} < \boldsymbol{\theta}^{\mathrm{L}}$, and $\mathbf{y}^{k} > \boldsymbol{\theta}^{\mathrm{U}}$, for some infeasible points. Therefore, the number of the constitutive box constraints becomes $\mathrm{N}^{\mathrm{Bnds}} = 2\mathrm{D}$ whenever our algorithm utilizes the constitutive box constraints in order to capture bounds if the data implies (give clues to) the presence of such constraints. Thus, 2D parameters (decision variables) are added to the optimization problem for complete set of the constitutive bound constraints.

These box constraints, when written in the generic form $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{j}})$ given above, take the forms given as:

$$\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Z}, \theta_{\mathrm{j}}) \coloneqq +\mathbf{z}_{\mathrm{i}} - \theta_{\mathrm{i}}^{\mathrm{U}}; \qquad \mathrm{i} = 1, \dots, \mathrm{D}; \quad \mathrm{j} = 1, \dots, \mathrm{D}, \tag{5.1a}$$

$$\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Z}, \theta_{\mathrm{j}}) \coloneqq -\mathbf{z}_{\mathrm{i}} + \theta_{\mathrm{i}}^{\mathrm{L}}; \qquad \mathrm{i} = 1, \dots, \mathrm{D}; \quad \mathrm{j} = 1 + \mathrm{D}, \dots, 2\mathrm{D}. \tag{5.1b}$$

For the feasible points, i.e., $\mathbf{z}_{\mathrm{i}}$ is replaced with $\mathbf{x}_{\mathrm{i}}$, the constitutive constraint is treated as $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{X}, \boldsymbol{\theta}_{\mathrm{j}}) \leq \mathbf{0}$. For the infeasible points, i.e., $\mathbf{z}_{\mathrm{i}}$ is replaced with $\mathbf{y}_{\mathrm{i}}$, the direction

of the inequalities must be changed, and the constitutive constraint is treated as $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Y}, \boldsymbol{\theta}_{\mathrm{j}}) > \mathbf{0}$.

A constitutive linear inequality constraint is written with respect to each feasible point, $\mathbf{x}^{l}$, (each row of $\mathbf{X}$) as: $\boldsymbol{\theta}_{\mathrm{c}}[\mathbf{x}^{l}]^{\mathrm{T}} \leq \theta_0$, and with respect to each infeasible point, $\mathbf{y}^{k}$, (each row of $\mathbf{Y}$) as: $\boldsymbol{\theta}_{\mathrm{c}}[\mathbf{y}^{k}]^{\mathrm{T}} > \theta_0$ where the $1 \times \mathrm{D}$ vector $\boldsymbol{\theta}_{\mathrm{c}}$ includes coefficients (slope parameters) of each of dimension and θ_0 value is the intercept parameter. Additionally, the transpose of each feasible and infeasible points, i.e., $[\mathbf{x}^{l}]^{\mathrm{T}}$ and $[\mathbf{y}^{k}]^{\mathrm{T}}$ are the $\mathrm{D} \times 1$ column vectors. The task of identifying an unknown linear inequality constraint with its generic form results in $\mathrm{D} + 1$ unknown parameters. Therefore, $\mathrm{N}^{\mathrm{Lin}}(\mathrm{D} + 1)$ parameters are added to the optimization problem for complete set of the constitutive linear inequality constraints when our algorithm utilizes $\mathrm{N}^{\mathrm{Lin}}$ such constraints.

These linear constraints, when written in the generic form $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{j}})$ given above, take the form given as:

$$\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{j}}) \coloneqq \sum_{\mathrm{i}=1}^{\mathrm{D}} (\theta_{\mathrm{ji}} \mathbf{z}_{\mathrm{i}}) - \theta_{\mathrm{j0}} \; ; \quad \mathrm{j} = 1, \dots, \mathrm{N}^{\mathrm{Lin}}. \tag{5.2}$$

For the feasible points, i.e., $\mathbf{z}_{\mathrm{i}}$ is replaced with $\mathbf{x}_{\mathrm{i}}$, the constitutive constraint is treated as $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{X}, \boldsymbol{\theta}_{\mathrm{j}}) \leq \mathbf{0}$. For the infeasible points, i.e., $\mathbf{z}_{\mathrm{i}}$ is replaced with $\mathbf{y}_{\mathrm{i}}$, the direction of the inequalities of infeasible points must be changed and the constitutive constraint for infeasible points is treated as $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Y}, \boldsymbol{\theta}_{\mathrm{j}}) > \mathbf{0}$.

A constitutive circular inequality constraint can also be represented with respect to each feasible point, $\mathbf{x}^{l}$, (each row of $\mathbf{X}$) as: $[\mathbf{x}^{l} - \boldsymbol{\theta}_{\mathrm{c}}][\mathbf{x}^{l} - \boldsymbol{\theta}_{\mathrm{c}}]^{\mathrm{T}} \leq \theta_0$ and with respect to each infeasible point, $\mathbf{y}^{k}$, (each row of $\mathbf{Y}$) as: $[\mathbf{y}^{k} - \boldsymbol{\theta}_{\mathrm{c}}][\mathbf{y}^{k} - \boldsymbol{\theta}_{\mathrm{c}}]^{\mathrm{T}} > \theta_0$ where $1 \times \mathrm{D}$ vector $\boldsymbol{\theta}_{\mathrm{c}}$ determines the coordinates of the centre of the circular feasible or infeasible region in each dimension and θ_0 value is the squared radius of feasible or infeasible circular region. Like the constitutive linear inequality constraints, $\mathrm{D} + 1$ parameters are required per constitutive circular inequality constraint. Thus, $\mathrm{N}^{\mathrm{Circ}}(\mathrm{D} + 1)$ parameters are added to the optimization problem for the complete set of the constitutive circular inequality constraints when our algorithm employs $\mathrm{N}^{\mathrm{Circ}}$ such constraints.

These circular constraints, when written in the generic form $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{j}})$ given above, take the form given as:

$$\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{j}}) \coloneqq \sum_{\mathrm{I}=1}^{\mathrm{D}} (\mathbf{z}_{\mathrm{i}} - \theta_{\mathrm{ji}})^2 - \theta_{\mathrm{j0}}; \quad \mathrm{j} = 1, \dots, \mathrm{N}^{\mathrm{Circ}}. \tag{5.3}$$

For the feasible points, i.e., $\mathbf{z}_{\mathrm{i}}$ is replaced with $\mathbf{x}_{\mathrm{i}}$, the constitutive constraint is treated as $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{X}, \boldsymbol{\theta}_{\mathrm{j}}) \leq \mathbf{0}$. For the infeasible points, i.e., $\mathbf{z}_{\mathrm{i}}$ is replaced with $\mathbf{x}_{\mathrm{i}}$, the constitutive constraint is treated as $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{X}, \boldsymbol{\theta}_{\mathrm{j}}) \leq \mathbf{0}$. For the infeasible points, i.e., $\mathbf{z}_{\mathrm{i}}$ is replaced with $\mathbf{y}_{\mathrm{i}}$, the direction of the inequalities must be changed, and the constitutive constraint is treated as $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Y}, \boldsymbol{\theta}_{\mathrm{j}}) > \mathbf{0}$.

Lastly, likewise the previous constitutive constraints, the constitutive ellipsoidal inequality constraint with covariance matrix can be demonstrated with respect to each feasible point, $\mathbf{x}^l$, (each row of $\mathbf{X}$) as: $[\mathbf{x}^l - \boldsymbol{\theta}_{\mathrm{c}}]\boldsymbol{\theta}_{\mathrm{M}}^{-1}[\mathbf{x}^l - \boldsymbol{\theta}_{\mathrm{c}}]^{\mathrm{T}} \leq \theta_0$, and with respect to each infeasible point, $\mathbf{y}^k$, (each row of $\mathbf{Y}$) as: $[\mathbf{y}^k - \boldsymbol{\theta}_{\mathrm{c}}]\boldsymbol{\theta}_{\mathrm{M}}^{-1}[\mathbf{y}^k - \boldsymbol{\theta}_{\mathrm{c}}]^{\mathrm{T}} > \theta_0$ where $1 \times \mathrm{D}$ vector $\boldsymbol{\theta}_{\mathrm{c}}$ defines the coordinates of the centre of the ellipsoidal inequality constraint in each dimension, the inverse of $\mathrm{D} \times \mathrm{D}$ square symmetric covariance matrix $\boldsymbol{\theta}_{\mathrm{M}}^{-1}$ encompasses information about elongation and rotation of the ellipse, and the θ_0 value is the parameter of the squared radius of the feasible or infeasible ellipsoidal region. It should be noted that rather than the covariance matrix itself, in this thesis work, its inverse is directly parameterized in order to avoid CPU-intensive matrix inversion at each function evaluation during optimization for every ellipsoidal constitutive constraint. The inverse of a symmetric matrix is also a symmetric matrix. It can be easily concluded that the deployment of single constitutive ellipsoidal inequality constraints results in $((\mathrm{D}^2 + \mathrm{D})/2 + \mathrm{D} + 1)$ parameters. Therefore, $\mathrm{N}^{\mathrm{Elps}}\,(\mathrm{D}^2 + 3\mathrm{D} + 2)/2$ parameters are added to the optimization problem for the complete set of the constitutive ellipsoidal inequality constraints by the deployment of $\mathrm{N}^{\mathrm{Elps}}$ such constraints.

These constitutive ellipsoidal constraints, when written in the generic form $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{j}})$ given above, take the form given as (where N is either N^{f} or N^{i}):

$$\hat{\mathrm{g}}_{\mathrm{j}}(\mathbf{z}^l, \boldsymbol{\theta}_{\mathrm{j}}) \coloneqq [\mathbf{z}^l - \boldsymbol{\theta}_{\mathrm{jc}}]\boldsymbol{\theta}_{\mathrm{jM}}^{-1}[\mathbf{z}^l - \boldsymbol{\theta}_{\mathrm{jc}}]^{\mathrm{T}} - \theta_{\mathrm{j0}}; \quad l = 1, \dots, \mathrm{N}; \quad \mathrm{j} = 1, \dots \mathrm{N}^{\mathrm{Elps}}, \tag{5.4a}$$

$$\hat{\mathbf{g}}_j(\mathbf{Z}, \boldsymbol{\theta}_j) \coloneqq \left[\hat{g}_j(\mathbf{z}^1, \boldsymbol{\theta}_j) \quad \cdots \quad \hat{g}_j(\mathbf{z}^l, \boldsymbol{\theta}_j) \quad \cdots \quad \hat{g}_j(\mathbf{z}^N, \boldsymbol{\theta}_j)\right]^T. \tag{5.4b}$$

For each feasible point, i.e., $\mathbf{z}^l$ is replaced with $\mathbf{x}^l$, the constitutive constraint is treated as $\hat{\mathbf{g}}_j(\mathbf{X}, \boldsymbol{\theta}_j) \leq \mathbf{0}$. For each infeasible point, i.e., $\mathbf{z}^l$ is replaced with $\mathbf{y}^k$, the direction of the inequalities must be changed, and the constitutive constraint is treated as $\hat{\mathbf{g}}_j(\mathbf{Y}, \boldsymbol{\theta}_j) > \mathbf{0}$.

Although this formulation is vectorized over N points (either N^f feasible or N^i infeasible points) for the evaluation of a single ellipsoidal constraint, our algorithm turns out to be too slow with increasing number of points and increasing number of such constraints, and thus, requires quite high CPU times. This issue instigated us to seek and develop a different method of constraint identification. Therefore, we will introduce this new method (the design-matrix approach) and generalize all constitutive-constraint forms in Chapter 7.

It should also be noted that the set of feasible and infeasible points, $\mathbf{X}$ and $\mathbf{Y}$, can directly be replaced with the boundary set of feasible and infeasible points, $\mathbf{X}^b$ and $\mathbf{Y}^b$, in the above paragraphs if the reduced sets are utilized in constraint identification. In such cases, N, N^f, and N^i must also be replaced with N^b, N^{fb}, and N^{ib}, respectively.

These constitutive inequality constraints can be exemplified explicitly in Table 5.1, for a two-dimensional case, i.e., with $N^f = 1$ and $D = 2$.

By looking at Equation (5.5a) and Equation (5.5b), the readers can easily observe that constraint-identification algorithm can detect bound constraints by deploying only the constitutive linear inequality constraints by zeroing either θ_1 or θ_2 during optimization. However, the utilization of a constitutive linear inequality constraint introduces three parameters (optimization decision variables) instead of the single parameter of the pure bound constraint.

Table 5.1. Explicit mathematical representations of constitutive inequality constraints for the two-dimensional case.

Type of Inequality Constraint	**Parametric Form of Inequality Constraint** $\hat{g}(z_1, z_2, \boldsymbol{\theta})$	**Eq. No**
Bounds	$+z_1 - \theta_1^U \leq 0,$ $-z_1 + \theta_1^L \leq 0,$ $+z_2 - \theta_2^U \leq 0,$ $-z_2 + \theta_2^L \leq 0,$	(5.5a)
Linear	$\theta_1 z_1 + \theta_2 z_2 - \theta_0 \leq 0,$	(5.5b)
Circular	$(z_1 - \theta_1)^2 + (z_2 - \theta_2)^2 - \theta_0 \leq 0,$	(5.5c)
Ellipsoidal	$\theta_3(z_1 - \theta_1)^2 + \theta_4(z_2 - \theta_2)^2 + 2\theta_5(z_1 - \theta_1)(z_2 - \theta_2) - \theta_0 \leq 0,$	(5.5d)
	where $\begin{cases} \boldsymbol{\theta}_M^{-1} = \begin{bmatrix} \theta_3 & \theta_5 \\ \theta_5 & \theta_4 \end{bmatrix} \text{ and } \boldsymbol{\theta}_c = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}^T. \end{cases}$	

Similarly, by looking at Equation (5.5c) and Equation (5.5d), readers can see that an ellipsoidal inequality constraint already embraces a circular inequality constraint. In other words, with the deployment of constitutive ellipsoidal inequality constraint, it is possible to fit a circular inequality constraint as well. For instance, as can be seen in Equation (5.5d), the parameters θ_3 and θ_4 are the diagonal elements of covariance matrix and the parameter θ_5 is the off-diagonal element of the symmetric covariance matrix. If the off-diagonal element(s), θ_5, becomes zero and the diagonal elements, θ_3 and θ_4, become identical, this generic ellipsoidal formulation will become identical to the generic circular constraint. Additionally, if the off-diagonal element(s), θ_5, is kept at zero but the diagonal elements, θ_3 and θ_4, become unequal to each other, it is possible to obtain elongated circular inequality constraint (vertical or horizontal ellipsoid). The off-diagonal element(s) determines whether cross terms exist or not. If the off-diagonal element(s) does not equal zero, cross terms prevail, and the constitutive ellipsoidal constraint gives the constraint-identification algorithm the ability to fit rotated ellipsoidal constraints. Moreover, the number of cross terms for D-dimensional point is equal to $\binom{D}{2} = D!/\big((D-2)!\,2!\big)$ when a

constitutive ellipsoidal inequality constraint is written explicitly for D-dimensional problem. For instance, with $D = 2$ the number of cross terms becomes one, i.e., x_1x_2. With $D = 3$ the number of cross terms becomes three, i.e., x_1x_2, x_1x_3 and x_2x_3. This topic will be scrutinized in Chapter 7.

5.2. Constraint Aggregation and the Objective Function

Constraint-identification algorithm depends on the feasibility or infeasibility of the sample points. However, the feasible region, which includes the feasible points and excludes the infeasible points, cannot be determined by considering the feasibility of the constitutive inequality constraints individually (e.g., via constraint-by-constraint feasibility check) as discussed and demonstrated in Chapter 2. Therefore, we need to embed all N^{Type} constitutive inequality constraints into a single function (a single embedding inequality constraint) which guarantees the feasibility of all the individual constitutive inequality constraints if and only if this single function, when evaluated with the feasible points, shows feasibility and, when evaluated with the infeasible points, shows infeasibility. With the KS aggregation function, which was thoroughly discussed in Chapter 4, we can obtain one single aggregated formulation of all the constitutive constraints. Constraint Aggregation (CA) via the KS function describes the feasible region as the union of all constitutive constraints. Thus, the values of all N^{Type} constitutive inequality constraints evaluated at the feasible and infeasible points can be reduced to the values of a single aggregated inequality constraint via CA, which will be shown in general form as $\mathbb{C}(\mathbf{Z}, \boldsymbol{\theta})$ by:

$$\mathbb{C}(\mathbf{Z}, \boldsymbol{\theta}) \coloneqq \overline{\mathrm{KS}}_{\mathrm{U}}\,(\mathbf{Z}, \boldsymbol{\theta}, \rho), \tag{5.6}$$

where $\mathbb{C}(.)$ is the aggregation operator which embeds N^{Type} constitutive inequality constraints into a single mathematical form using the KS function. $\mathbb{C}(\mathbf{X}, \boldsymbol{\theta})$ denotes the values of a single aggregated function of constitutive constraints evaluated at the feasible points, $\mathbf{X}$, with the current values of parameters, $\boldsymbol{\theta}$, of all N^{Type} constitutive inequality constraints when $\mathbf{Z}$ is replaced with $\mathbf{X}$. $\mathbb{C}(\mathbf{Y}, \boldsymbol{\theta})$ implies the values of a single aggregated function of constitutive constraints evaluated at the infeasible points, $\mathbf{Y}$, with the current values of parameters, $\boldsymbol{\theta}$, of all N^{Type} constitutive inequality constraints when $\mathbf{Z}$ is replaced with $\mathbf{Y}$. It is also noted that the row cardinalities of $\mathbb{C}(\mathbf{X}, \boldsymbol{\theta})$ and $\mathbb{C}(\mathbf{Y}, \boldsymbol{\theta})$ adhere to those of

$\hat{\mathbf{g}}_j(\mathbf{X}, \boldsymbol{\theta}_j)$ and $\hat{\mathbf{g}}_j(\mathbf{Y}, \boldsymbol{\theta}_j)$, i.e., $\mathbb{C}(\mathbf{X}, \boldsymbol{\theta})$ is the $N^f \times 1$ and $\mathbb{C}(\mathbf{Y}, \boldsymbol{\theta})$ is the $N^i \times 1$ (i.e, what are aggregated are the constitutive constraints, not the set of feasible or infeasible points).

With the optimal values of the parameters of the N^{Type} constitutive inequality constraints, the values of all the N^{Type} constitutive inequality constraints evaluated at feasible points, $\mathbf{X}$, should be less than or equal to zero, i.e., $\hat{\mathbf{g}}_j(\mathbf{X}, \boldsymbol{\theta}_j) \leq \mathbf{0}$, and thus their aggregation, i.e., $\mathbb{C}(\mathbf{X}, \boldsymbol{\theta}) \leq \mathbf{0}$, as explicitly shown as:

$$\mathbb{C}(\mathbf{X}, \boldsymbol{\theta}) \coloneqq \hat{\mathbf{g}}_{max} + \frac{1}{\rho} \ln \left(\sum_{j=1}^{N^{Type}} e^{\rho(\hat{\mathbf{g}}_j(\mathbf{X}, \boldsymbol{\theta}_j) - \hat{\mathbf{g}}_{max})} \right) \leq \mathbf{0}, \tag{5.7a}$$

$$\mathbb{C}(\mathbf{Y}, \boldsymbol{\theta}) \coloneqq \hat{\mathbf{g}}_{max} + \frac{1}{\rho} \ln \left(\sum_{j=1}^{N^{Type}} e^{\rho(\hat{\mathbf{g}}_j(\mathbf{Y}, \boldsymbol{\theta}_j) - \hat{\mathbf{g}}_{max})} \right) > \mathbf{0}, \tag{5.7b}$$

where $\hat{\mathbf{g}}_{max} = \max_j \{\hat{\mathbf{g}}_j(\mathbf{X}, \boldsymbol{\theta}_j)\}$. However, even though the outcomes of some N^{Type} constitutive inequality constraints evaluated at some infeasible points, $\mathbf{Y}$, may be less than zero, i.e., $\hat{\mathbf{g}}_j(\mathbf{Y}, \boldsymbol{\theta}_j) < \mathbf{0}$, the outcomes of their aggregation should be greater than zero, i.e., $\mathbb{C}(\mathbf{Y}, \boldsymbol{\theta}) > \mathbf{0}$ which is explicitly shown in Equation (5.7b) where $\hat{\mathbf{g}}_{max} = \max_j \{\hat{\mathbf{g}}_j(\mathbf{Y}, \boldsymbol{\theta}_j)\}$.

The overall objective function and its sub-parts, which will be explained thoroughly, are all given by:

$$\min_{\boldsymbol{\theta}} \; J_1(\mathbf{X}, \mathbf{Y}, \boldsymbol{\theta}) = \sum_{l=1}^{N^f} \max\{0, \mathbb{C}(\mathbf{x}^l, \boldsymbol{\theta})\} + \sum_{k=1}^{N^i} \max\{0, -\mathbb{C}(\mathbf{y}^k, \boldsymbol{\theta})\}, \tag{5.8a}$$

$$\min_{\boldsymbol{\theta}} \; J_2(\mathbf{X}, \mathbf{Y}, \boldsymbol{\theta}) = \text{count}\{\mathbb{C}(\mathbf{X}, \boldsymbol{\theta}) > \mathbf{0}\} + \text{count}\{\mathbb{C}(\mathbf{Y}, \boldsymbol{\theta}) \leq \mathbf{0}\}, \tag{5.8b}$$

$$\min_{\boldsymbol{\theta}} \; J_3(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1, \tag{5.8c}$$

$$\min_{\boldsymbol{\theta}} \; J(\mathbf{X}, \mathbf{Y}, \boldsymbol{\theta}) = J_1(\mathbf{X}, \mathbf{Y}, \boldsymbol{\theta}) + J_2(\mathbf{X}, \mathbf{Y}, \boldsymbol{\theta}) + \mu J_3(\boldsymbol{\theta}). \tag{5.8d}$$

Therefore, we can detect the violation of the aggregated constitutive constraints evaluated at each one of the feasible and infeasible points using the max{.} operator shown in Equation (5.8a). If the outcomes of the aggregation when evaluated at feasible points, $\mathbf{X}$, with the current values of $\boldsymbol{\theta}$, i.e., $\mathbb{C}(\mathbf{X}, \boldsymbol{\theta})$, (which is $N^f \times 1$ column vector) includes any positive value (which should not) for any feasible point, $\mathbf{x}^l$, this positive value is reflected

into the objective function (increases the objective function value under minimization). For infeasible points, if the negative of the outcome of the aggregation when evaluated at infeasible points, $\mathbf{Y}$, with the current values of $\boldsymbol{\theta}$, i.e., $-\mathbb{C}(\mathbf{Y}, \boldsymbol{\theta})$, (which is $\mathrm{N}^{\mathrm{i}} \times 1$ column vector) contains any positive value (which should not) for any infeasible point, $\mathbf{y}^k$, this positive value is reflected into the objective function (increases the objective function value under minimization) Therefore, the value of the first part, J_1 (shown in Equation (5.8a)), of the overall objective function, J (shown in Equation (5.8d)), should equal to zero at the optimum to make sure that each one of the constitutive inequality constraints with the optimal values of $\boldsymbol{\theta}$ satisfy all the feasible points and dissatisfy all the infeasible points.

However, when only this first part, J_1, of the objective function alone is utilized, the optimizer can reduce the value of this objective function towards quite small values up to the machine-epsilon levels. Although such a low objective value can be considered enough for many optimization problems, for some complicated problems with many constitutive constraints it may results in erroneous identification of the constitutive inequality constraints since the constitutive inequality constraints actually may not completely satisfy all the feasible and infeasible points. To resolve this issue, we introduce the second part, J_2 (given in Equation (5.8b)), of the overall objective function, J. By employing this function together with J_1, we can penalize the value of objective function with the "number count" of the violated points even if J_1 evaluates to a quite small value. Therefore, we can guarantee to find correct optimum parameters, $\boldsymbol{\theta}$, of the identified constitutive inequality constraints. The optimizer zeroing the J_2 also guarantees that the value of J_1 approaches to zero properly as conceived.

For some problems, possibly with redundant constitutive constraints, the parameters, $\boldsymbol{\theta}$, may not be unique. For example, one can multiply both sides of a linear inequality with a large constant and thus increase the values of all the parameters, $\boldsymbol{\theta}$, of the inequality constraints by this constant. However, original inequality constraints and the inequality constraints multiplied by a constant show the same identical feasible region boundary. Thus, lastly, to get the minimum-norm solution for the values of $\boldsymbol{\theta}$, we introduce third part, J_3 (given in Equation (5.8c)), of the overall objective function, J, where $\|.\|_1$ denotes the 1-norm operator. In other words, this part of the objective function forces the optimizer to find the values of the optimal parameters as small as possible. In addition to this, the

selection of 1-norm operator (as opposed to 2-norm e.g.) imparts sparsity of the optimal parameter vector. Enhancing sparsity of the solution can be useful to detect parameters whose actual values are equal to zero.

The overall objective function which will be utilized for the purpose of identifying the inequality constraints is given in Equation (5.8d) where μ denotes the weighting parameter of third sub-objective function. We will set the values of μ to something between 10^{-8} and 10^{-4}, depending on the problem.

5.3. Optimization Algorithm Used

The nature of the objective function introduced in this chapter does not allow us to employ a derivative-based optimizer (e.g., MATLAB's "fmincon" or "fminunc" algorithms) since the objective function given in Equation (5.8d) is, first of all, a non-smooth function. J_1 is a non-smooth function and is non-differentiable at zero due to the max{.} operator. J_2 is not continuous and non-differentiable due to "integer counting" operator. J_3 is also a non-smooth and non-differentiable at zero since 1-norm operator accommodates absolute value operator. Therefore, the derivative-based optimizers fail to reach global or even a good local optimum with this objective function unless it is completely smoothed (smoothing introduces approximation). Even if the objective function is completely smoothed, the derivative-based optimizers can be easily trapped into one of many local optima since the objective function is not convex and the objective function hypersurface contains uncountable and unpredictable local minima. Therefore, the non-smooth and non-convex characters of our objective function cause premature (nonoptimal) termination of the constraint-identification tasks.

On the other hand, there are many types of derivative-free stochastic optimization methods based on function evaluations only. These methods, also called "global optimization methods", are much more successful at avoiding local minima and approaching the global minimum compared to derivative-based optimizers. Various optimization methods which fall under this umbrella were tested and the Differential Evolution (DE) (Price et al., 2005) was found to be the most successful optimizer in the context of this thesis work.

DE is regarded as one of the stochastic global optimization methods. It is a population-based optimization algorithm. DE tries to optimize by keeping candidate solutions having a pre-determined population size. DE algorithm differs from other evolutionary algorithms (such as genetic algorithms) in its specific mutation and crossover procedures. DE algorithm includes four basic parts. These parts are initialization, mutation, crossover, and selection. For each optimization iteration three of them are executed except for initialization step. The first parameter of DE, which should be determined suitably before optimization, is the population size “NP”. The population size determines how many population vectors exist. In other words, this indicates that there are NP vectors of decision variables during the optimization. Initial values of NP vectors of decision variables are generally selected via generating uniformly distributed random numbers with pre-specified lower and upper bounds. This step is called initialization and only occurs at the first iteration. After this step, to form mutation vectors or new vectors of the decision variables, mutation parameter (F) is employed in tandem with a base vector and the difference between any two target vectors randomly selected. In other words, mutation vector is obtained as the combination of the base vector and the difference vector which is scaled with F. There are various mutation procedures based on some combination rules of existing candidate solutions to create new candidate solutions. Some mutation procedures also utilize an existing vector which results in the lowest function value (best target vector for related iteration) as a base vector for mutation. Additionally, some of them also include two difference vectors instead of one, and thus with such mutation procedures the creation of mutation vectors depends on base vector, two difference vectors and F. With the crossover parameter (CR), each of variables of a mutated vector (new candidate solution) and a target vector (existing candidate solution) is compared to determine which variables from these vectors are in corresponding trial vector. This comparison depends on specific rules, some of which utilize CR as threshold for the creation of the trial vectors. In other words, some of variables of a trial vector come from corresponding target vector and others come from the corresponding mutation vector. Additionally, the creation of trial vectors can be also done by combining related target vector with the difference vector between corresponding mutated vector and this target vector. Thus, the diversity enhancement is also provided with CR. The combination of a certain mutation procedure and a certain crossover procedure is called “strategy”. In the selection step, comparison among a trial vector and corresponding target vector is made based on the values of

function evaluated with these vectors. The vector which gives the lowest function value is preserved in the next generation or the next iteration as a target vector and other is discarded (Ahmad et al., 2022; Price et al., 2005).

In this thesis work, we employ a DE algorithm modified by Prof. Akman. This modified algorithm includes 10 strategies available in the original DE code (Price et al., 2005). However, these strategies are randomly selected during each iteration of DE. 13 different formulas are used for the determination of the values of F and CR, and these are also selected randomly during each iteration of DE. Additionally, this modified optimization algorithm enables optimizer to use adaptive population sizes during the optimization that depends on the standard deviation of the objective function values corresponding to population members. Thus, the DE algorithm modified by Prof. Akman is a (randomly) variable strategy, variable F and CR generation and adaptive population version of the original DE algorithm.

6. CONSTRAINT IDENTIFICATION EXAMPLES

In this chapter, we will present constraint-identification examples by employing basic constitutive inequality constraints introduced in Chapter 5. Firstly, we will begin presenting the examples with two-, three-, four- and eight-dimensional single circular inequality constraints and proceed with single elliptical, multiple bound, and multiple linear constraints, all of which yielding convex feasible regions. The last example of this section will be a two-dimensional nonconvex case that contains combination of some of the previous constitutive constraints. We will also compare the utilization of the complete sets and the boundary sets using the same constitutive inequality constraints for two- and three-dimensional scenarios.

6.1. Single Circular Constraint Identification

6.1.1. 2-D Case

In this sub-section, we will demonstrate the ability of our algorithm to identify a two-dimensional circular inequality constraint, i.e., $D = 2$. The circular inequality constraint which will be treated as unknown yet will be detected by our algorithm is given by the following equation:

$$g_1: \ (z_1 - 4)^2 + (z_2 - 2.5)^2 - 10 \leq 0. \tag{6.1}$$

Meshwise sampling (MS) is employed to generate the sample points. The cardinality of the set of sample points is $N = 2500$. The lower and upper bounds on the sample points are specified as $\mathbf{z}^{LB} = [-1 \ -2]$ and $\mathbf{z}^{UB} = [+9 \ +7]$. The number of constitutive circular inequality constraint is $N^{Circ} = 1$. Although the value of ρ is specified as 500, there is no effect on the constraint-identification task since only one constitutive constraint is present. The complete set of points, **Z**, is a 2500×2 matrix. The sets of feasible, **X**, and infeasible, **Y**, points, generated by the sampling procedure (mentioned in Chapter 3), are 832×2 and 1668×2 matrices respectively, i.e., $N^f = 832$ and $N^i = 1668$.

The number of parameters (optimization decision variables) becomes $|\boldsymbol{\theta}| = 3$ for $N^{Circ} = 1$ and $D = 2$. The values of lower bounds are set to -10 and the values of upper bounds are set to +10 for each of the decision variables and the initial population size is taken as $NP = 40$. Our algorithm utilizes the objective function as the combination of J_1 and J_3 by excluding the second part, J_2, of the overall objective function, J, for this sub-section. Therefore, we will demonstrate that the utilization of only J_1 and J_3 as the overall objective function also works for such simple constraint-identification tasks. The value of the weighting parameter, μ, of third sub-objective function, J_3, is set to 10^{-4}.

The optimization terminates at the function tolerance of 10^{-9} after 1159 iterations with 32652 function evaluations and gives the optimal minimum value of the overall objective function as $J = 16.47 \times 10^{-4}$. The value of the first part of the objective function becomes zero. i.e., $J_1 = 0$, while the value of the third part of the objective function becomes $J_3 = 16.47$. Even though J_2 is discarded from the overall objective function for this problem, the second sub-objective function (violations-count function), evaluated at the optimal values of parameters of this constitutive inequality constraint, is found to be zero as expected.

The constitutive circular inequality constraint identified is given explicitly together with the optimal values of parameters as:

$$\hat{g}(\boldsymbol{\theta}) = (z_1 - 4.000)^2 + (z_2 - 2.500)^2 - 9.969. \tag{6.2}$$

Figure 6.1 presents the identified constitutive circular constraint over the region formed by the sets of feasible (blue) and infeasible (red) points. By comparing the identified constitutive circular inequality constraint given by Equation (6.2) with the actual circular inequality constraint given in Equation (6.1), it can be concluded that our algorithm identifies the actual circular inequality constraint successfully.

It is also possible to observe from Figure 6.1 that the identified constitutive inequality constraint does not lead to any violations over the sets of feasible and infeasible points. In other words, the identified constraint includes all feasible points and excludes all infeasible points. The zero value of J_1 implies this as well.

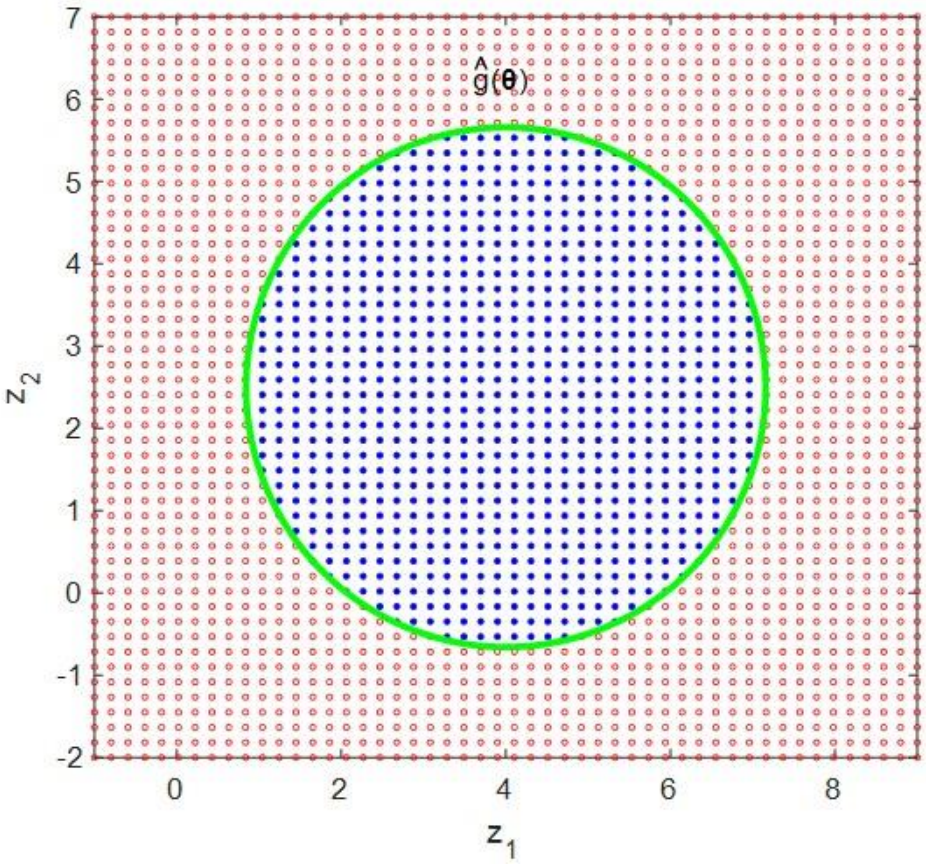

Figure 6.1. Identified optimal constitutive circular inequality constraint superimposed with the complete sets of feasible and infeasible points.

To compare the results obtained with the complete feasible and infeasible points and those obtained with the boundary-zone points (introduced in Chapter 3), we present the second identification example using the same settings given above. However, in this example, p^{th} percentile is specified as $p = 2$. Thus, the boundary sets of feasible, $\mathbf{X}^{b}$, and infeasible, $\mathbf{Y}^{b}$, points become 296×2 and 342×2 matrices respectively, i.e., $N^{fb} = 296$ and $N^{ib} = 342$.

The optimization terminates at the function tolerance of 10^{-9} after 817 iterations with 20274 function evaluations and gives the optimal value of the overall objective function as $J = 16.47 \times 10^{-4}$. The value of the first part of the objective function, J_1, becomes zero, while the value of the third part of the objective function becomes $J_3 = 16.47$. Even though J_2 is discarded from the overall objective function for this problem, the second sub-objective function, evaluated at the optimal values of the parameters, is also found to be zero as expected.

The constitutive circular inequality constraint identified is given explicitly together with the optimal values of parameters as:

$$\hat{g}(\boldsymbol{\theta}) = (z_1 - 4.000)^2 + (z_2 - 2.500)^2 - 9.969. \tag{6.3}$$

The constitutive circular inequality constraint identified is depicted in Figure 6.2. By comparing the constitutive circular inequality constraint given in Equation (6.3) to the actual circular inequality constraint given in Equation (6.1), it can be seen that our algorithm identified the actual inequality constraint equally successfully. Additionally, it is possible to observe from Figure 6.2 that the identified constraint does not cause any violation at the boundary and thus satisfies all the feasible and infeasible boundary points. It should be also highlighted that there is no difference between the utilization of the complete sets and the boundary sets except that our algorithm identifies the constraint with much less CPU time under the boundary-zone formation. Therefore, we will utilize the boundary sets, rather than the complete sets, to identify constraints for most parts of this chapter.

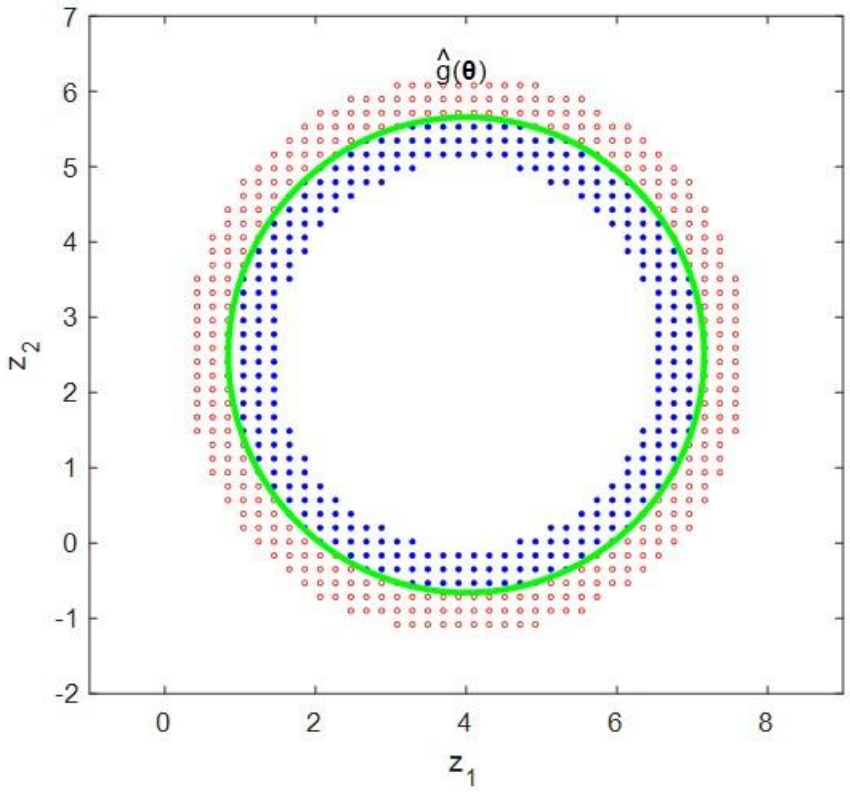

Figure 6.2. Identified optimal constitutive circular inequality constraint superimposed with the boundary sets of feasible and infeasible points.

6.1.2. 3-D Case

In this sub-section, we will demonstrate the ability of our algorithm to identify a three-dimensional circular (spherical) inequality constraint, i.e., $D = 3$. The spherical inequality constraint which will be treated as unknown and will be detected by our algorithm is given by the following equation:

$$g_1: \ (z_1 - 3.5)^2 + (z_2 - 2.5)^2 + (z_3 - 4)^2 - 12 \leq 0. \qquad (6.4)$$

Meshwise sampling (MS) is employed to generate the sample points. The cardinality of the set of sample points is $N = 2500$. The lower and upper bounds on the sample points are specified as $\mathbf{z}^{LB} = [-1 \ -2 \ -1]$ and $\mathbf{z}^{UB} = [+9 \ +7 + 8]$. The number of constitutive spherical inequality constraint is $N^{Circ} = 1$. Although the value of ρ is specified as 500, there is no effect on the constraint-identification task since only one constitutive constraint is present. The complete set of points, $\mathbf{Z}$, is a 2744×3 matrix. The sets of feasible, $\mathbf{X}$, and infeasible, $\mathbf{Y}$, points, generated by the sampling procedure (mentioned in Chapter 3), are 476×3 and 2268×3 matrices respectively, i.e., $N^f = 476$ and $N^i = 2268$.

The number of parameters (optimization decision variables) becomes $|\boldsymbol{\theta}| = 4$ for $N^{Circ} = 1$ and $D = 3$. The values of lower bounds are set to −20 and the values of upper bounds are set to +20 for each of the decision variables and the initial population size is taken as $NP = 40$.

Like the previous sub-section, our algorithm utilizes the objective function as the combination of J_1 and J_3 by excluding the second part, J_2, of the overall objective function, J, for this sub-section. Therefore, we will prove that the utilization of only J_1 and J_3 as the overall objective function also works for such simple constraint-identification tasks with $D = 3$. The value of the weighting parameter, μ, of third sub-objective function, J_3, is set to 10^{-4}.

The optimization terminates at the function tolerance of 10^{-9} after 2301 iterations with 61516 function evaluations and gives the optimal minimum value of the overall

objective function as $J = 21.92 \times 10^{-4}$. The value of the first part of the objective function becomes zero. i.e., $J_1 = 0$, while the value of the third part of the objective function becomes $J_3 = 21.92$. Even though J_2 is discarded from the overall objective function for this problem, the second sub-objective function (violations-count function), evaluated at the optimal values of parameters of this constitutive inequality constraint, is found to be zero as expected.

The constitutive spherical inequality constraint identified is given explicitly together with the optimal values of parameters as:

$$\hat{g}(\boldsymbol{\theta}) = (z_1 - 3.507)^2 + (z_2 - 2.500)^2 + (z_3 - 3.991)^2 - 11.926. \qquad (6.5)$$

Figure 6.3a presents the identified constitutive spherical constraint with the three-dimensional region formed by the sets of feasible (blue) and infeasible (red) points and Figure 6.3b-d show the projections of Figure 6.3a onto all pairs of binary planes.

By comparing the identified constitutive spherical inequality constraint given by Equation (6.5) with the actual spherical inequality constraint given in Equation (6.4), it can be concluded that our algorithm identifies the actual spherical inequality constraint successfully.

It is also possible to observe from Figure 6.3 that the identified constitutive inequality constraint, whose surface (boundary) is denoted by black spherical mesh (a cage that encloses all the blue feasible points), does not lead to any violations within this three-dimensional region. In other words, the identified constraint includes all feasible points and excludes all infeasible points. The zero value of J_1 implies this as well.

It should be added that the surface (boundary) of the identified constraint shown in Figure 6.3 is drawn by using MATLAB's "isosurface" function. This function provides the surface representation of three-dimensional points by connecting points of a constant value within a volume and can be regarded as a surface contour of three-dimensional points.

To compare the results obtained with the complete feasible and infeasible points and those obtained with the boundary-zone points (introduced in Chapter 3), we present the

second identification example using the same settings given above. However, in this example, p^{th} percentile is specified as $p = 2$.

Thus, the boundary sets of feasible, $\mathbf{X}^{b}$, and infeasible, $\mathbf{Y}^{b}$, points become 352×3 and 500×3 matrices respectively, i.e., $N^{fb} = 352$ and $N^{ib} = 500$.

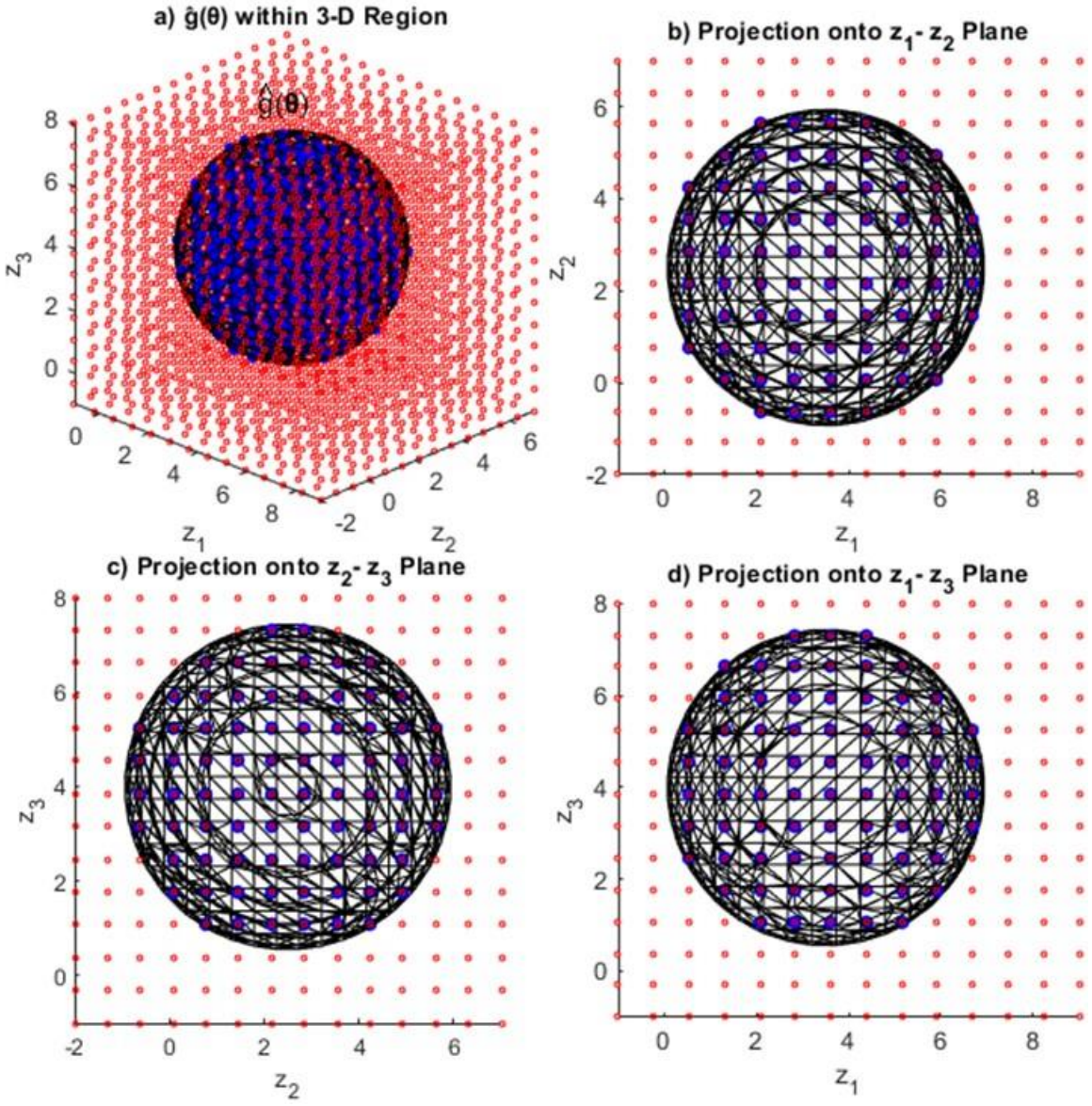

Figure 6.3. Identified optimal constitutive spherical inequality constraint superimposed with the complete sets of feasible and infeasible points.

The optimization terminates at the function tolerance of 10^{-9} after 2251 iterations with 57960 function evaluations and gives the optimal value of the overall objective function as $J = 21.92 \times 10^{-4}$.

The value of the first part of the objective function, J_1, becomes zero, while the value of the third part of the objective function becomes $J_3 = 21.92$.

Even though J_2 is discarded from the overall objective function for this problem, the second sub-objective function, evaluated at the optimal values of the parameters, is also found to be zero as expected.

The constitutive spherical inequality constraint identified is given explicitly together with the optimal values of parameters as:

$$\hat{g}(\boldsymbol{\theta}) = (z_1 - 3.507)^2 + (z_2 - 2.500)^2 + (z_3 - 3.991)^2 - 11.926. \qquad (6.6)$$

Figure 6.4a presents the identified constitutive spherical constraint with the three-dimensional region formed by the sets of feasible (blue) and infeasible (red) points and Figure 6.4b-d show the projections of Figure 6.4a onto all pairs of binary planes.

By comparing the constitutive spherical inequality constraint given in Equation (6.6) to the actual spherical inequality constraint given in Equation (6.4), it can be seen that our algorithm identifies the actual inequality constraint equally successfully.

Additionally, it is possible to observe from Figure 6.4 that the identified constraint does not cause any violation at the boundary and thus satisfies all the feasible and infeasible boundary points.

It should be also highlighted that there is no difference between the utilization of the complete sets and the boundary sets except that our algorithm identifies the constraint with much less CPU time under the boundary-zone formation except that our algorithm identifies the constraint with much less CPU time under the boundary-zone formation.

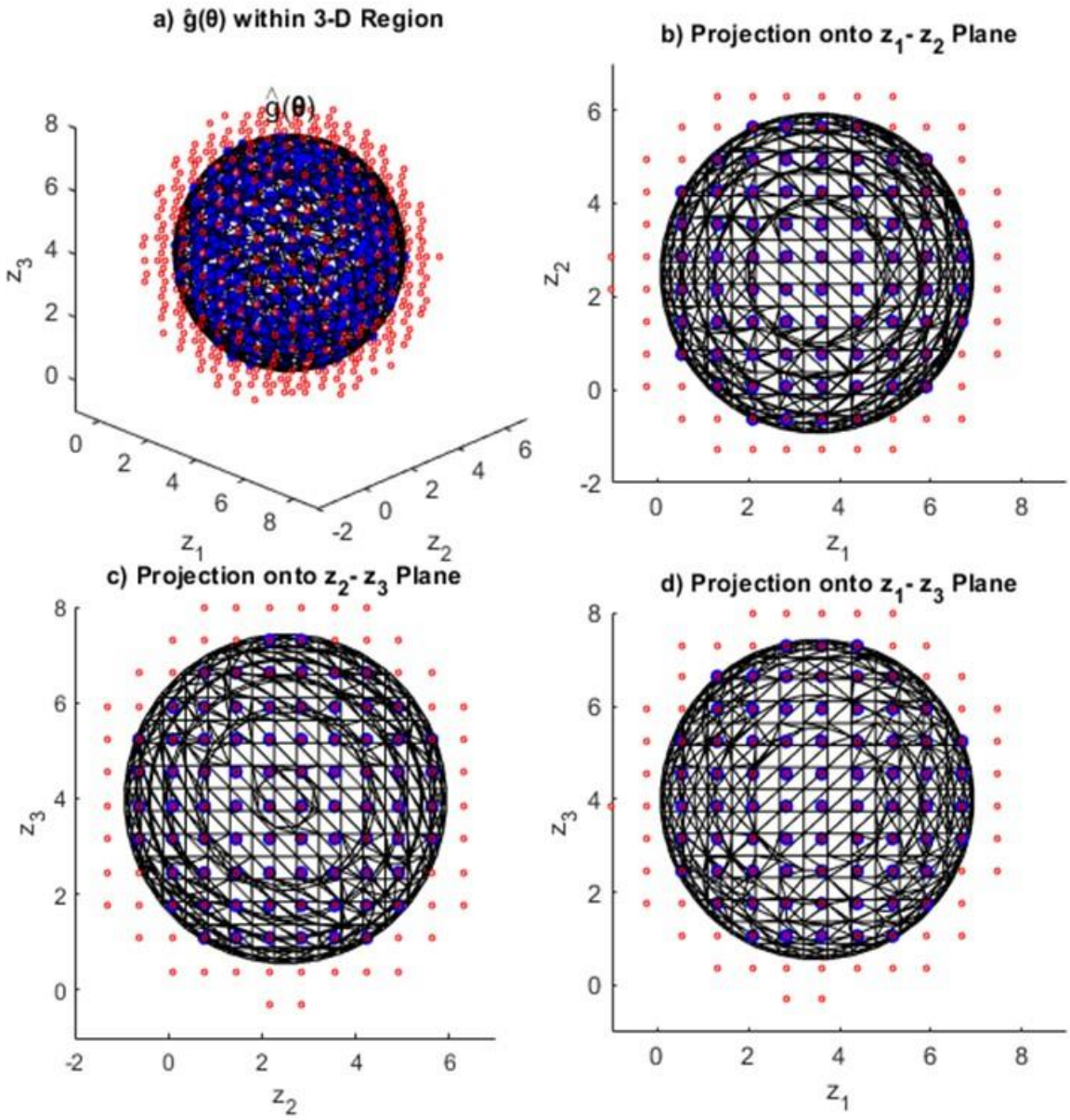

Figure 6.4. Identified optimal constitutive spherical inequality constraint superimposed with the boundary sets of feasible and infeasible points.

6.1.3. n-D Case

In this sub-section, we will demonstrate the ability of our algorithm to identify higher-dimensional circular (hyperspherical) inequality constraints, i.e., $D = 4$ and $D = 8$. The four-dimensional circular inequality constraint which will be treated as unknown and will be detected by our algorithm is given by the following equation:

$$g_1: \ (z_1 - 6.4)^2 + (z_2 - 2.5)^2 + (z_3 - 5)^2 + (z_4 - 8)^2 - 20 \leq 0. \qquad (6.7)$$

Stratified Latin Hypercube Sampling (SLHS) is employed to generate the sample points for this high-dimensional case. The cardinality of the set of sample points is $N = 2500$. The lower and upper bounds on the sample points are specified as $\mathbf{z}^{LB} = [-1\ -2\ -2\ -1]$ and $\mathbf{z}^{UB} = [+11\ +7\ +8\ +12]$. The number of constitutive circular inequality constraint is $N^{Circ} = 1$. Although the value of ρ is specified as 500, there is no effect on the constraint-identification task since only one constitutive constraint is present. The complete set of points, **Z**, is a 4096×4 matrix. The sets of feasible, **X**, and infeasible, **Y**, points, generated by the sampling procedure (mentioned in Chapter 3), are 536×4 and 3560×4 matrices respectively, i.e., $N^{f} = 536$ and $N^{i} = 3560$. In this example, p^{th} percentile is specified as $p = 1$. Thus, the boundary sets of feasible, $\mathbf{X}^{b}$, and infeasible, $\mathbf{Y}^{b}$, points become 427×4 and 611×4 matrices respectively, i.e., $N^{fb} = 427$ and $N^{ib} = 611$.

The number of parameters (optimization decision variables) becomes $|\boldsymbol{\theta}| = 5$ for $N^{Circ} = 1$ and $D = 4$. The values of lower bounds are set to −50 and the values of upper bounds are set to +50 for each of the decision variables and the initial population size is taken as $NP = 40$. Our algorithm utilizes the objective function as the combination of J_1 and J_3 by excluding the second part, J_2, of the overall objective function, J, for this sub-section. Therefore, we will demonstrate that the utilization of only J_1 and J_3 as the overall objective function also works for such simple constraint-identification tasks with $D = 4$. The value of the weighting parameter, μ, of third sub-objective function, J_3, is set to 10^{-4}.

The optimization terminates at the function tolerance of 10^{-9} after 11151 iterations with 161307 function evaluations and gives the optimal minimum value of the overall objective function as $J = 41.83 \times 10^{-4}$. The value of the first part of the objective function becomes zero. i.e., $J_1 = 0$, while the value of the third part of the objective function becomes $J_3 = 41.83$.

Even though J_2 is discarded from the overall objective function for this problem, the second sub-objective function (violations-count function), evaluated at the optimal values of parameters of this constitutive inequality constraint, is found to be zero as expected.

The constitutive four-dimensional circular inequality constraint identified is given explicitly together with the optimal values of parameters as:

$$\hat{g}(\boldsymbol{\theta}) = (z_1 - 6.407)^2 + (z_2 - 2.477)^2 + (z_3 - 5.006)^2 + (z_4 - 7.984)^2 - 19.960. \tag{6.8}$$

By comparing the identified constitutive circular inequality constraint given by Equation (6.8) with the actual circular inequality constraint given in Equation (6.7), it can be concluded that our algorithm identifies the actual circular inequality constraint successfully. Although there is no chance to visually assess whether the identified constitutive circular constraint satisfies all the feasible and infeasible points, the zero value of J_1 implies that the identified constitutive circular constraint does not lead to any violations of the feasible and infeasible points.

The eight-dimensional circular inequality constraint which will be treated as unknown and will be detected by our algorithm is given by the following equation:

$$g_1 : (z_1 - 2.7)^2 + (z_2 - 3)^2 + (z_3 - 4)^2 + (z_4 - 1)^2 + (z_5 - 5)^2 + (z_6 - 1.5)^2 + (z_7 - 7)^2 + (z_8 - 5)^2 - 35 \leq 0. \tag{6.9}$$

SLHS is employed again to generate the sample points. The cardinality of the set of sample points is $N = 5000$. The lower and upper bounds on the sample points are specified as $\mathbf{z}^{LB} = [-1 - 2 - 1 - 2 - 2 - 1 - 2 - 2]$ and $\mathbf{z}^{UB} = [+6 + 5 + 9 + 5 + 9 + 5 + 10 + 8]$. The number of constitutive circular inequality constraint is $N^{Circ} = 1$. The complete set of points, $\mathbf{Z}$, is a 6561×8 matrix. The sets of feasible, $\mathbf{X}$, and infeasible, $\mathbf{Y}$, points, generated by the sampling procedure (mentioned in Chapter 3), are 788×8 and 5773×8 matrices respectively, i.e., $N^f = 788$ and $N^i = 5773$. In this example, p^{th} percentile is specified as $p = 1$. Thus, the boundary sets of feasible, $\mathbf{X}^b$, and infeasible, $\mathbf{Y}^b$, points become 773×8 and 1685×8 matrices respectively, i.e., $N^{fb} = 773$ and $N^{ib} = 1685$.

The number of parameters (optimization decision variables) becomes $|\boldsymbol{\theta}| = 9$ for $N^{Circ} = 1$ and $D = 8$. The values of lower bounds are set to −50 and the values of upper bounds are set to +50 for each of the decision variables and the initial population size is

taken as $NP = 40$. Our algorithm utilizes the objective function as the combination of J_1 and J_3 by excluding the second part, J_2, of the overall objective function, J, for this sub-section. Therefore, we will demonstrate that the utilization of only J_1 and J_3 as the overall objective function also works for such simple constraint-identification tasks with $D = 8$. The value of the weighting parameter, μ, of third sub-objective function, J_3, is set to 10^{-4}.

The optimization terminates at the function tolerance of 10^{-9} after 36551 iterations with 921261 function evaluations and gives the optimal minimum value of the overall objective function as $J = 64.01 \times 10^{-4}$. The value of the first part of the objective function becomes zero. i.e., $J_1 = 0$, while the value of the third part of the objective function becomes $J_3 = 64.01$.

Even though J_2 is discarded from the overall objective function for this problem, the second sub-objective function (violations-count function), evaluated at the optimal values of parameters of this constitutive inequality constraint, is found to be zero as expected.

The constitutive eight-dimensional circular inequality constraint identified is given explicitly together with the optimal values of parameters as:

$$\begin{aligned}\hat{g}(\boldsymbol{\theta}) = \; & (z_1 - 2.696)^2 + (z_2 - 2.976)^2 + (z_3 - 4.003)^2 \\ & + (z_4 - 0.987)^2 + (z_5 - 5.007)^2 + (z_6 - 1.501)^2 \\ & + (z_7 - 6.994)^2 + (z_8 - 4.982)^2 - 34.860.\end{aligned} \tag{6.10}$$

By comparing the identified constitutive circular inequality constraint given by Equation (6.10) with the actual circular inequality constraint given in Equation (6.9), it can be concluded that our algorithm identifies the actual eight-dimensional circular inequality constraint successfully.

Although there is no chance to assess whether the identified constitutive circular constraint satisfies all the feasible and infeasible points in a visual manner, the zero value of J_1 implies that the identified constitutive circular constraint does not lead to any violations of the feasible and infeasible points.

6.2. Single Ellipsoidal Constraint Identification

6.2.1. 2-D Case

In this sub-section, we will demonstrate the ability of our algorithm to identify a two-dimensional ellipsoidal inequality constraint, i.e., $D = 2$. The ellipsoidal inequality constraint which will be treated as unknown and will be detected by our algorithm is given by the following equation:

$$g_1: \ (z_1 - 4)^2 + 2(z_2 - 2.5)^2 - 2(z_1 - 4)(z_2 - 2.5) - 6 \leq 0. \qquad (6.11)$$

Meshwise sampling (MS) is employed to generate the sample points. The cardinality of the set of sample points is $N = 2500$. The lower and upper bounds on the sample points are specified as $\mathbf{z}^{LB} = [-1 \ -2]$ and $\mathbf{z}^{UB} = [+9 \ +7]$. The number of constitutive ellipsoidal inequality constraint is $N^{Elps} = 1$. Although the value of ρ is specified as 500, there is no effect on the constraint-identification task since only one constitutive constraint is present. The complete set of points, **Z**, is a 2500×2 matrix. The sets of feasible, **X**, and infeasible, **Y**, points, generated by the sampling procedure (mentioned in Chapter 3), are 502×2 and 1998×2 matrices respectively, i.e., $N^f = 502$ and $N^i = 1998$. In this example, p^{th} percentile is specified as $p = 1$. Thus, the boundary sets of feasible, $\mathbf{X}^b$, and infeasible, $\mathbf{Y}^b$, points become 164×2 and 179×2 matrices respectively, i.e., $N^{fb} = 164$ and $N^{ib} = 179$.

The number of parameters (optimization decision variables) becomes $|\boldsymbol{\theta}| = 6$ for $N^{Elps} = 1$ and $D = 2$. The values of lower bounds are set to −10 and the values of upper bounds are set to +10 for each of the decision variables and the initial population size is taken as $NP = 40$. Our algorithm utilizes all parts of the overall objective function introduced in Chapter 5. The value of the weighting parameter, μ, of third sub-objective function, J_3, is set to 10^{-7}.

The optimization terminates at the function tolerance of 10^{-9} after 9001 iterations with 326370 function evaluations and gives the optimal minimum value of the overall objective function as $J = 6.491 \times 10^{-7}$. The values of the first and second parts of the

objective function become zero. i.e., $J_1 = 0$ and $J_2 = 0$, while the value of the third part of the objective function becomes $J_3 = 6.491$.

The constitutive two-dimensional ellipsoidal inequality constraint identified is given explicitly together with the optimal values of parameters by:

$$\begin{aligned}\hat{g}(\boldsymbol{\theta}) = {} & 0.003(z_1 - 3.978)^2 + 0.006(z_2 - 2.481)^2 \\ & - 0.006(z_1 - 3.978)(z_2 - 2.481) - 0.018,\end{aligned} \tag{6.12a}$$

$$\begin{aligned}\frac{\hat{g}(\boldsymbol{\theta})}{0.003} = {} & (z_1 - 3.978)^2 + 1.999(z_2 - 2.481)^2 \\ & - 1.982(z_1 - 3.978)(z_2 - 2.481) - 6.005,\end{aligned} \tag{6.12b}$$

$$\begin{aligned}\boldsymbol{\theta}_M^{-1} &= \begin{bmatrix}\theta_3 & \theta_5 \\ \theta_5 & \theta_4\end{bmatrix} \\ &= \begin{bmatrix}1 & -0.991 \\ -0.991 & 1.999\end{bmatrix},\end{aligned} \tag{6.12c}$$

$$\boldsymbol{\theta}_M = \begin{bmatrix}1.965 & 0.974 \\ 0.974 & 0.983\end{bmatrix}. \tag{6.12d}$$

Additionally, Equation (6.12b) shows the identified constraint normalized by the first parameter, 0.003. They are thus identical and both represent the same feasible-region boundary, as mentioned in Chapter 5. The inverse of the covariance matrix associated with the ellipse (see Chapter 5, Equation (5.5d)) is given by Equation (6.12c) and the covariance matrix itself by Equation (6.12d).

The unequal diagonal elements of the covariance matrix indicate that the elongation (principle axis) is in z_1-direction, and the positive sign of the off-diagonal elements indicates that the ellipse is rotated clockwise.

Figure 6.5 presents the identified constitutive ellipsoidal constraint over the region formed by the boundary sets of feasible (blue) and infeasible (red) points. By comparing the identified constitutive ellipsoidal inequality constraint given by Equation (6.12b) with the actual ellipsoidal inequality constraint given in Equation (6.11), it can be concluded that our algorithm identifies the actual ellipsoidal inequality constraint successfully.

It is also possible to observe from Figure 6.5 that the identified constitutive inequality constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible boundary points. The zero values of $J_1$and J_2 imply this as well.

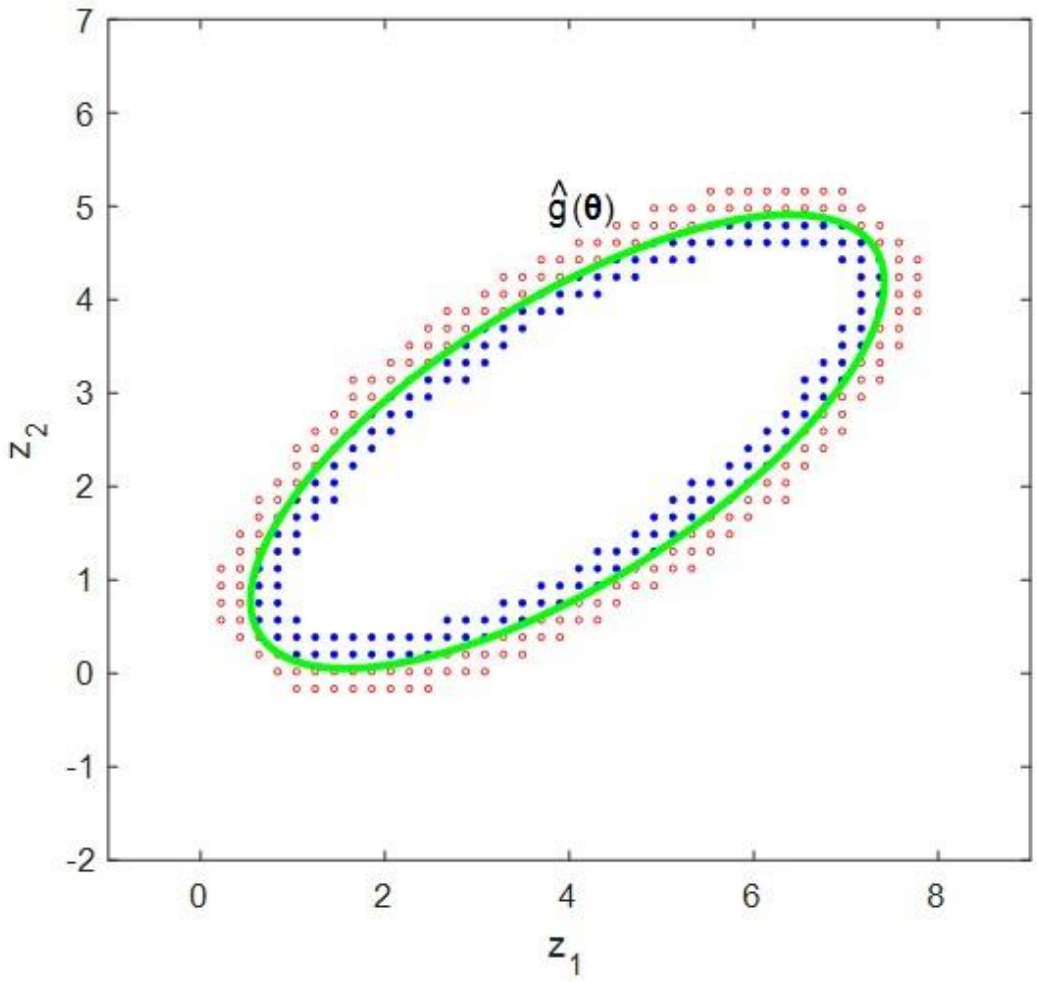

Figure 6.5. Identified optimal constitutive ellipsoidal inequality constraint superimposed with the boundary sets of feasible and infeasible points.

6.2.2. 3-D Case

In this sub-section, we will demonstrate the ability of our algorithm to identify a three-dimensional ellipsoidal (hyperellipsoid) inequality constraint, i.e., $D = 3$. The ellipsoidal inequality constraint which will be treated as unknown and will be detected by our algorithm is given by the following equation:

$$\begin{aligned} g_1:\ & 1.5(z_1-4)^2 + 3(z_2-2.5)^2 + 1.5(z_3-5)^2 + (z_1-4)(z_2-2.5) \\ & + 2(z_1-4)(z_3-5) + 1.5(z_2-2.5)(z_3-5) - 15 \leq 0. \end{aligned} \tag{6.13}$$

Meshwise sampling (MS) is employed to generate the sample points. The cardinality of the set of sample points is $N = 5000$. The lower and upper bounds on the sample points are specified as $\mathbf{z}^{LB} = [-1 \quad -2 \quad 0]$ and $\mathbf{z}^{UB} = [+9 \quad +7 \quad +10]$. The number of constitutive spherical inequality constraint is $N^{Elps} = 1$. Although the value of ρ is specified as 500, there is no effect on the constraint-identification task since only one constitutive constraint is present.

The complete set of points, **Z**, is a 5832×3 matrix. The sets of feasible, **X**, and infeasible, **Y**, points, generated by the sampling procedure (mentioned in Chapter 3), are 724×3 and 5108×3 matrices respectively, i.e., $N^{f} = 724$ and $N^{i} = 5108$. In this example, p^{th} percentile is specified as $p = 0.3$. Thus, the boundary sets of feasible, $\mathbf{X}^{b}$, and infeasible, $\mathbf{Y}^{b}$, points become 336×3 and 409×3 matrices respectively, i.e., $N^{fb} = 336$ and $N^{ib} = 409$.

The number of parameters (optimization decision variables) becomes $|\boldsymbol{\theta}| = 10$ for $N^{Elps} = 1$ and $D = 3$. The values of lower bounds are set to −20 and the values of upper bounds are set to +20 for each of the decision variables and the initial population size is taken as $NP = 40$. Our algorithm utilizes all parts of the overall objective function. The value of the weighting parameter, μ, of third sub-objective function, J_3, is set to 10^{-7}.

The optimization terminates at the function tolerance of 10^{-9} after 72451 iterations with 2556624 function evaluations and gives the optimal minimum value of the overall objective function as $J = 11.73 \times 10^{-7}$.

The values of the first and second parts of the objective function becomes zero. i.e., $J_1 = 0$ and $J_2 = 0$, while the value of the third part of the objective function becomes $J_3 = 11.73$.

The constitutive three-dimensional ellipsoidal inequality constraint identified is given explicitly together with the optimal values of parameters as:

$$\begin{aligned}\hat{g}(\boldsymbol{\theta}) = {} & 0.0157(z_1 - 3.998)^2 + 0.031(z_2 - 2.495)^2 \\ & + 0.0156(z_3 - 4.995)^2 \\ & + 0.0104(z_1 - 3.998)(z_2 - 2.495) \\ & + 0.0208(z_1 - 3.998)(z_3 - 4.995) \\ & + 0.0158(z_2 - 2.495)(z_3 - 4.995) - 0.156,\end{aligned} \tag{6.14a}$$

$$\begin{aligned}\frac{\hat{g}(\boldsymbol{\theta})}{0.0104} = {} & 1.486(z_1 - 3.998)^2 + 2.939(z_2 - 2.495)^2 \\ & + 1.485(z_3 - 4.995)^2 + 1(z_1 - 3.998)(z_2 - 2.495) \\ & + 1.981(z_1 - 3.998)(z_3 - 4.995) \\ & + 1.515(z_2 - 2.495)(z_3 - 4.995) - 14.812.\end{aligned} \tag{6.14b}$$

Additionally, Equation (6.14b) shows the identified constraint normalized by 0.0104. They are thus identical and both represent the same feasible-region boundary as mentioned in Chapter 5

Figure 6.6a presents the identified constitutive spherical constraint with the three-dimensional region formed by the boundary sets of feasible (blue) and infeasible (red) points and Figure 6.6b-d show the projections of Figure 6.6a onto all pairs of its binary planes.

By comparing the identified constitutive ellipsoidal inequality constraint given by Equation (6.14b) with the actual ellipsoidal inequality constraint given in Equation (6.13), it can be concluded that our algorithm identifies the actual ellipsoidal inequality constraint successfully.

It is also possible to observe from Figure 6.6 that the identified constitutive inequality constraint, whose surface (boundary) is denoted by black spherical mesh (a cage that encloses all the blue feasible points), does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible boundary points. The zero values of J_1 and J_2 implies this as well.

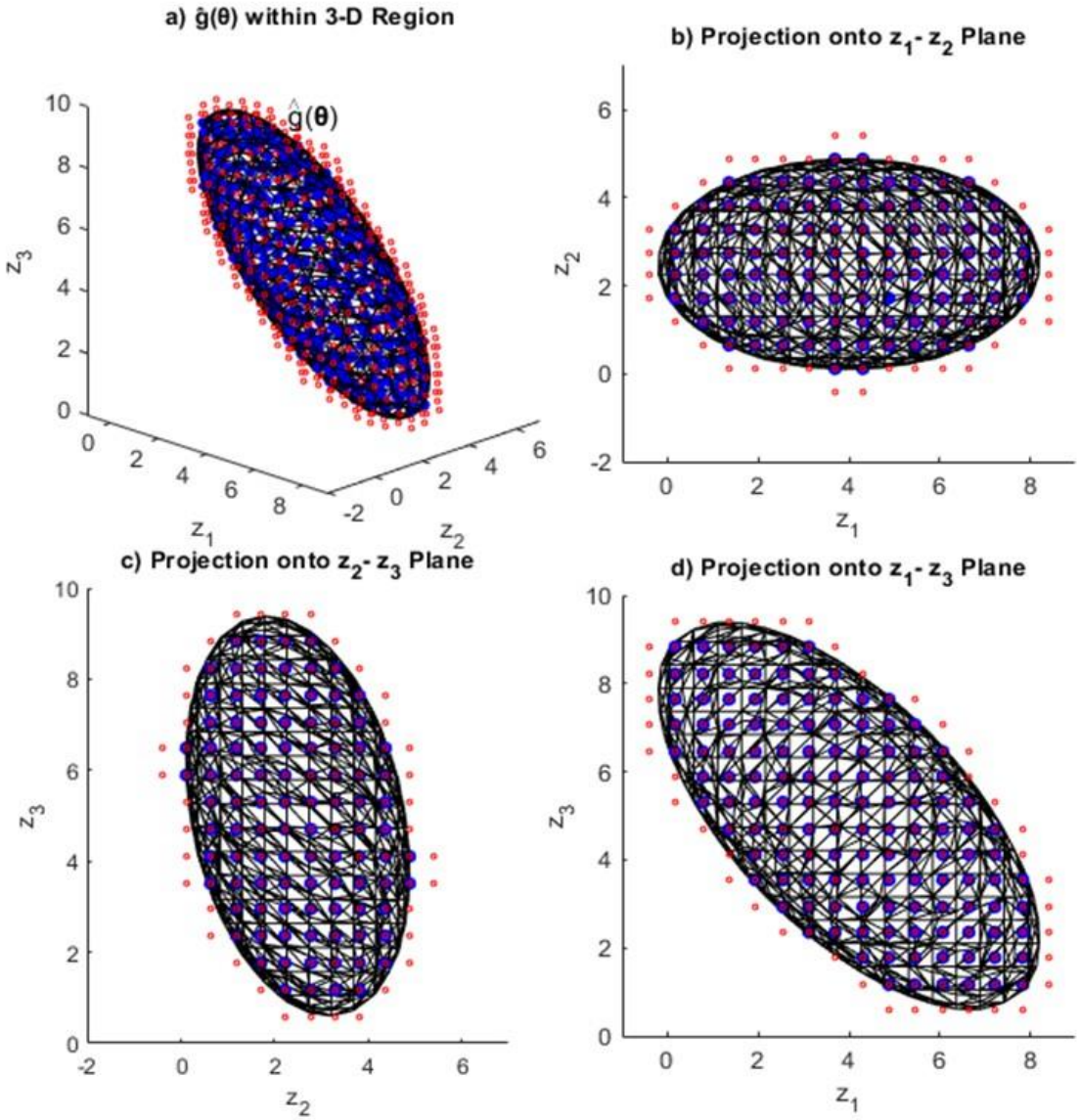

Figure 6.6. Identified optimal constitutive ellipsoidal inequality constraint superimposed with the boundary sets of feasible and infeasible points.

6.3. Bound Constraints Identification

6.3.1. 2-D Case

In this sub-section, we will demonstrate the ability of our algorithm to identify bound (box) constraints with two-dimensional region, i.e., $D = 2$. The box constraints which will be treated as unknown and will be detected by our algorithm are given by the following equation:

$$g_1: \quad +z_1 - 7 \leq 0, \tag{6.15a}$$

$$g_2: \quad -z_1 + 1 \leq 0, \tag{6.15b}$$

$$g_3: \quad +z_2 - 5 \leq 0, \tag{6.15c}$$

$$g_4: \quad -z_2 \leq 0. \tag{6.15d}$$

SLHS is employed to generate the sample points. The cardinality of the set of sample points is $N = 2500$. The lower and upper bounds on the sample points are specified as $\mathbf{z}^{LB} = [-1 \ -2]$ and $\mathbf{z}^{UB} = [+9 \ +7]$. The number of constitutive box constraints, which depends on the dimension, is $N^{Bnds} = 2D = 4$. The value of ρ is specified as 500. The complete set of points, **Z**, is a 2500×2 matrix. The sets of feasible, **X**, and infeasible, **Y**, points, generated by the sampling procedure (mentioned in Chapter 3), are 833×2 and 1667×2 matrices respectively, i.e., $N^f = 833$ and $N^i = 1667$. In this example, p^{th} percentile is specified as $p = 1$. Thus, the boundary sets of feasible, $\mathbf{X}^b$, and infeasible, $\mathbf{Y}^b$, points become 239×2 and 260×2 matrices respectively, i.e., $N^{fb} = 239$ and $N^{ib} = 260$

The number of parameters (optimization decision variables) becomes $|\boldsymbol{\theta}| = 4$ for $D = 2$ and $N^{Bnds} = 4$. The values of lower bounds are set to −10 and the values of upper bounds are set to +10 for each of the decision variables and the initial population size is taken as $NP = 40$. Our algorithm utilizes all parts of the overall objective function introduced in Chapter 5. The value of the weighting parameter, μ, of third sub-objective function, J_3, is set to 10^{-4}.

The optimization terminates at the function tolerance of 10^{-9} after 324 iterations with 12668 function evaluations and gives the minimum value of the overall objective function as $J = 12.99 \times 10^{-4}$. The values of the first and second parts of the objective function become zero. i.e., $J_1 = 0$ and $J_2 = 0$, while the value of the third part of the objective function becomes $J_3 = 12.99$.

The constitutive bound constraints identified are given explicitly together with the optimal values of parameters in:

$$\hat{g}_1(\theta_1) = +z_1 - 6.996, \tag{6.16a}$$

$$\hat{g}_2(\theta_2) = -z_1 + 0.997, \tag{6.16b}$$

$$\hat{g}_3(\theta_3) = +z_2 - 4.999, \tag{6.16c}$$

$$\hat{g}_4(\theta_4) = -z_2 + 0.000. \tag{6.16d}$$

Figure 6.7 presents the aggregated constraint, $\mathbb{C}(\boldsymbol{\theta})$, by the light green rectangle and the identified individual constitutive bound constraints by the yellow, purple, dark green and cyan lines over the region formed by the boundary sets of feasible (blue) and infeasible (red) points. By comparing the identified constitutive bound constraints given by Equation (6.16) with the actual bound constraints given in Equation (6.15), it can be concluded that our algorithm identifies the actual bound constraints successfully. It is also possible to observe from Figure 6.7 that the aggregated constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible boundary points. The zero values of J_1 and J_2 imply this as well.

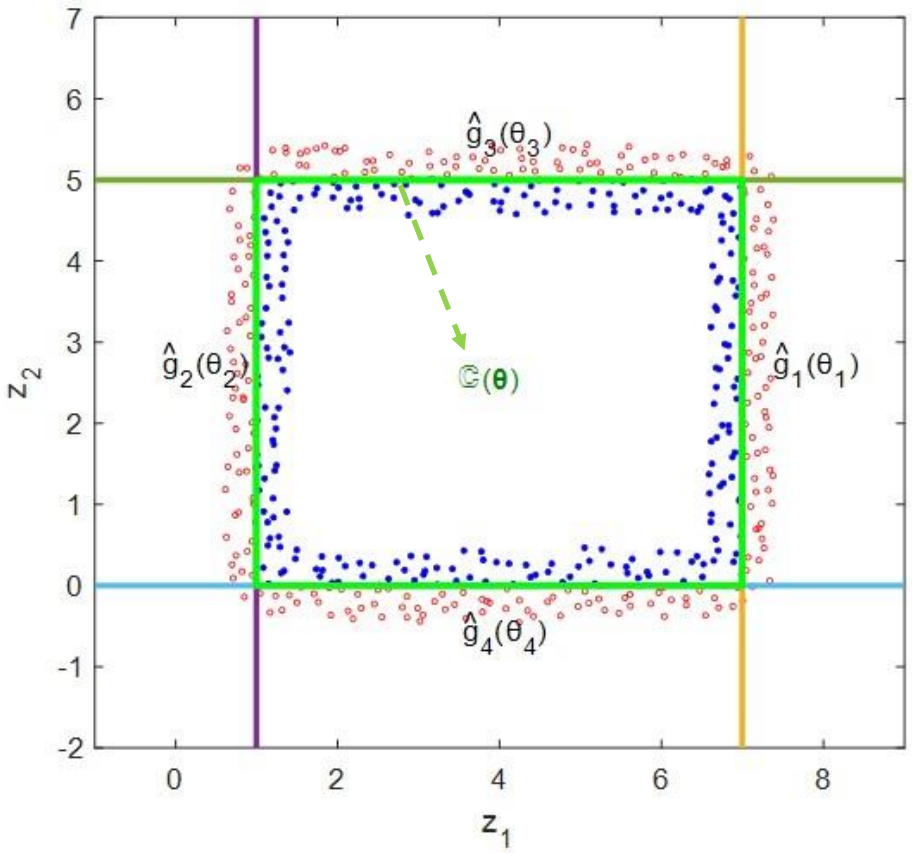

Figure 6.7. Identified optimal constitutive bound constraints and aggregated constraint superimposed with the boundary sets of feasible and infeasible points.

6.3.2. 3-D Case

In this sub-section, we will demonstrate the ability of our algorithm to identify bound constraints within three-dimensional region, i.e., $D = 3$. The bound constraints which will be treated as unknown and will be detected by our algorithm are given by the following equation:

$$g_1: \quad +z_1 - 8 \leq 0, \tag{6.17a}$$

$$g_2: \quad -z_1 + 1 \leq 0, \tag{6.17b}$$

$$g_3: \quad +z_2 - 5 \leq 0, \tag{6.17c}$$

$$g_4: \quad -z_2 \leq 0, \tag{6.17d}$$

$$g_5: \quad +z_3 - 6 \leq 0, \tag{6.17e}$$

$$g_6: \quad -z_3 - 1 \leq 0. \tag{6.17f}$$

SLHS is employed to generate the sample points. The cardinality of the set of sample points is fixed at $N = 5000$. The lower and upper bounds on the sample points are specified as $\mathbf{z}^{LB} = [-1 \ -2 \ -2]$ and $\mathbf{z}^{UB} = [+9 \ +7 \ +8]$. The number of constitutive box constraints, which depends on the dimension, is $N^{Bnds} = 2D = 6$. The value of ρ is specified as 500. The complete set of points, $\mathbf{Z}$, is a 5832×3 matrix.

The sets of feasible, $\mathbf{X}$, and infeasible, $\mathbf{Y}$, points, generated by the sampling procedure (mentioned in Chapter 3), are 1584×3 and 4248×3 matrices respectively, i.e., $N^f = 1584$ and $N^i = 4248$. In this example, p^{th} percentile is specified as $p = 0.5$. Thus, the boundary sets of feasible, $\mathbf{X}^b$, and infeasible, $\mathbf{Y}^b$, points become 885×3 and 1121×3 matrices respectively, i.e., $N^{fb} = 885$ and $N^{ib} = 1121$.

The number of parameters (optimization decision variables) becomes $|\boldsymbol{\theta}| = 6$ for $D = 3$ and $N^{Bnds} = 6$. The values of lower bounds are set to −10 and the values of upper bounds are set to +10 for each of the decision variables and the initial population size is taken as $NP = 40$. Our algorithm utilizes all parts of the overall objective function. The value of the weighting parameter, μ, of third sub-objective function, J_3, is set to 10^{-4}.

The optimization terminates at the function tolerance of 10^{-9} after 440 iterations with 14590 function evaluations and gives the optimal minimum value of the overall objective function as $J = 20.99 \times 10^{-4}$.

The values of the first and second parts of the objective function becomes zero. i.e., $J_1 = 0$ and $J_2 = 0$, while the value of the third part of the objective function becomes $J_3 = 20.99$.

The constitutive box constraints identified are given explicitly together with the optimal values of parameters as:

$$\hat{g}_1(\theta_1) = +z_1 - 7.999, \tag{6.18a}$$

$$\hat{g}_2(\theta_2) = -z_1 + 0.998, \tag{6.18b}$$

$$\hat{g}_3(\theta_3) = +z_2 - 4.999, \tag{6.18c}$$

$$\hat{g}_4(\theta_4) = -z_2 + 0.000, \tag{6.18d}$$

$$\hat{g}_5(\theta_5) = +z_3 - 5.996, \tag{6.18e}$$

$$\hat{g}_6(\theta_6) = -z_3 - 0.998. \tag{6.18f}$$

Figure 6.8 presents the aggregated three-dimensional constraint, whose surface (boundary) is denoted by black hyperrectangular mesh (a cage that encloses all the blue feasible points), and provides the identified constitutive box constraints respectively by the red, green, dark blue, cyan, yellow, magenta planes with the three-dimensional region formed by the boundary sets of feasible (blue) and infeasible (red) points and Figure 6.8b-d show the projections of Figure 6.8a onto all pairs of its binary planes.

By comparing the identified constitutive box constraints given by Equation (6.18) with the actual bound constraints given in Equation (6.17), it can be concluded that our algorithm identifies the actual bound constraints successfully. It is also possible to observe from Figure 6.8 that the aggregated inequality constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible boundary points. The zero values of J_1 and J_2 implies this as well.

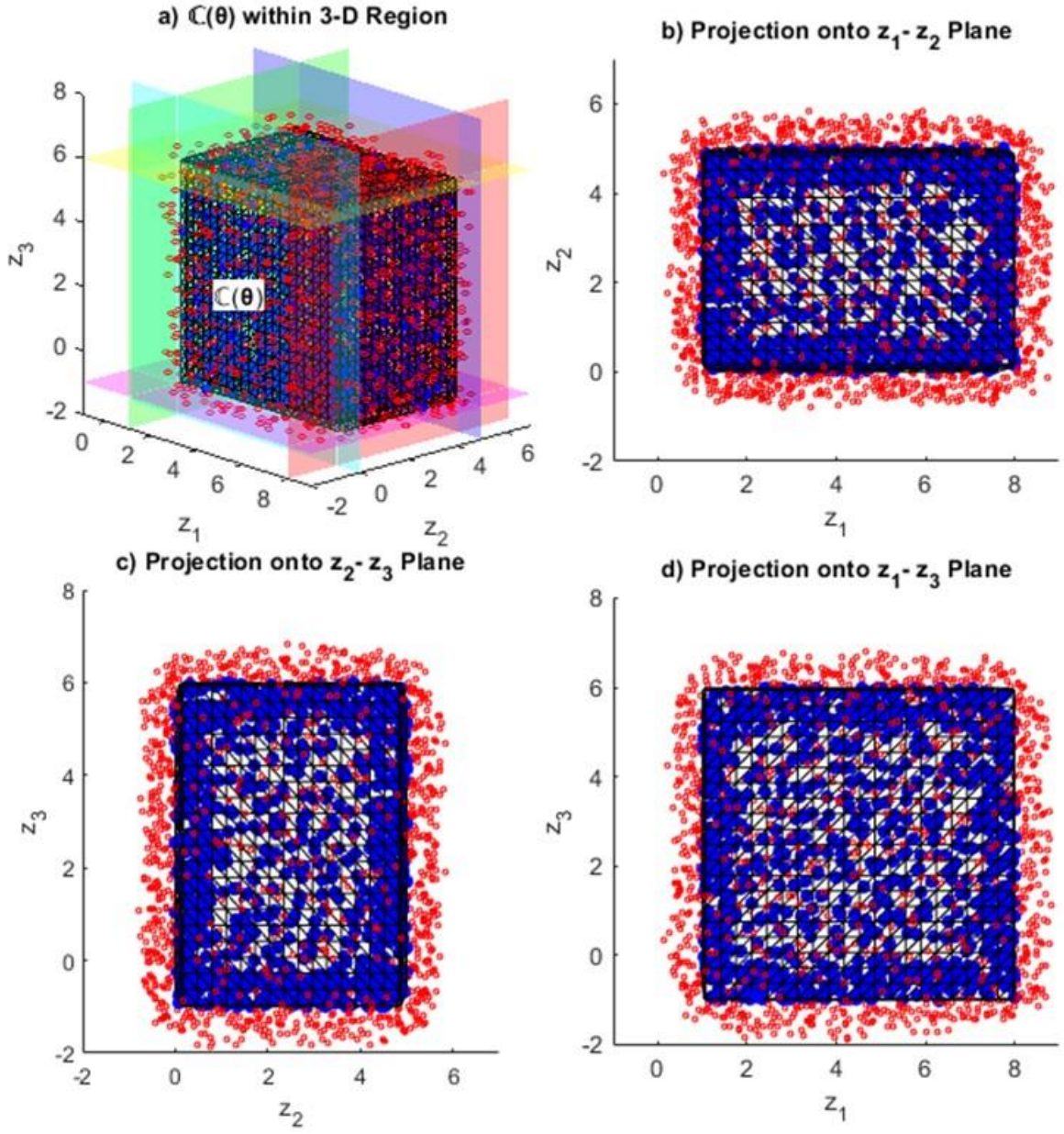

Figure 6.8. Identified optimal constitutive bound constraints and aggregated constraint superimposed with the boundary sets of feasible and infeasible points.

6.4. Identification of Linear Constraints

In this section, we will demonstrate the ability of our algorithm to identify two-dimensional mixture of linear inequality constraints, i.e., $D = 2$. Six linear inequality constraints which will be treated as unknowns and will be detected by our algorithm are given by the following equation:

$$g_1: \quad +z_1 - 7 \leq 0, \tag{6.19a}$$

$$g_2: \quad -z_1 + 1 \leq 0, \tag{6.19b}$$

$$g_3: \quad +z_2 - 5 \leq 0, \tag{6.19c}$$

$$g_4: \quad -z_2 \leq 0, \tag{6.19d}$$

$$g_5: \quad +z_1 + z_2 - 10 \leq 0, \tag{6.19e}$$

$$g_6: \quad -1.5z_1 - z_2 + 3.5 \leq 0. \tag{6.19f}$$

SLHS is employed to generate the sample points. The cardinality of the set of sample points is $N = 2500$. The lower and upper bounds on the sample points are specified as $\mathbf{z}^{LB} = [-1 \ -2]$ and $\mathbf{z}^{UB} = [+9 \ +7]$. The number of constitutive linear inequality constraints is $N^{Lin} = 6$. The value of ρ is specified as 500. The complete set of points, **Z**, is a 2500×2 matrix. The sets of feasible, **X**, and infeasible, **Y**, points, generated by the sampling procedure (mentioned in Chapter 3), are 739×2 and 1761×2 matrices respectively, i.e., $N^{f} = 739$ and $N^{i} = 1761$.

In this example, p^{th} percentile is specified as $p = 1$. Thus, the boundary sets of feasible, $\mathbf{X}^{b}$, and infeasible, $\mathbf{Y}^{b}$, points become 206×2 and 224×2 matrices respectively, i.e., $N^{fb} = 206$ and $N^{ib} = 224$.

The number of parameters (optimization decision variables) becomes $|\boldsymbol{\theta}| = 18$ for $N^{Lin} = 6$ and $D = 2$. The values of lower bounds are set to −10 and the values of upper bounds are set to +10 for each of the decision variables and the initial population size is taken as $NP = 40$. Our algorithm utilizes all parts of the overall objective function introduced in Chapter 5. The value of the weighting parameter, μ, of third sub-objective function, J_3, is set to 10^{-8}.

The optimization terminates at the function tolerance of 10^{-9} after 11276 iterations with 371467 function evaluations and gives the minimum value of the overall objective function as $J = 3.133 \times 10^{-7}$. The values of the first and second parts of the objective function become zero. i.e., $J_1 = 0$ and $J_2 = 0$, while the value of the third part of the objective function becomes $J_3 = 31.33$.

The constitutive linear inequality constraints identified are given explicitly together with the optimal values of parameters as:

$$\hat{g}_1(\boldsymbol{\theta}_1) = +0.018z_1 + 0.000z_2 - 0.132, \tag{6.20a}$$

$$\hat{g}_2(\boldsymbol{\theta}_2) = -0.081z_1 + 0.000z_2 + 0.081, \tag{6.20b}$$

$$\hat{g}_3(\boldsymbol{\theta}_3) = +0.000z_1 + 2.925z_2 - 14.617, \tag{6.20c}$$

$$\hat{g}_4(\boldsymbol{\theta}_4) = -0.000z_1 - 0.024z_2 - 0.000, \tag{6.20d}$$

$$\hat{g}_5(\boldsymbol{\theta}_5) = +0.752z_1 + 0.732z_2 - 7.443, \tag{6.20e}$$

$$\hat{g}_6(\boldsymbol{\theta}_6) = -1.133z_1 - 0.744z_2 + 2.642. \tag{6.20f}$$

Additionally, the identified constraints normalized by the related parameters are shown as:

$$\frac{\hat{g}_1(\boldsymbol{\theta}_1)}{0.018} = +1.000z_1 + 0.000z_2 - 7.018, \tag{6.21a}$$

$$\frac{\hat{g}_2(\boldsymbol{\theta}_2)}{0.081} = -1.000z_1 + 0.000z_2 + 1.000, \tag{6.21b}$$

$$\frac{\hat{g}_3(\boldsymbol{\theta}_3)}{2.925} = +0.000z_1 + 1.000z_2 - 4.997, \tag{6.21c}$$

$$\frac{\hat{g}_4(\boldsymbol{\theta}_4)}{0.024} = -0.000z_1 - 1.000z_2 - 0.000, \tag{6.21d}$$

$$\frac{\hat{g}_5(\boldsymbol{\theta}_5)}{0.752} = +1.000z_1 + 0.972z_2 - 9.886, \tag{6.21e}$$

$$\frac{\hat{g}_6(\boldsymbol{\theta}_6)}{0.744} = -1.522z_1 - 1.000z_2 + 3.551. \tag{6.21f}$$

Figure 6.9 presents the aggregated constraint, $\mathbb{C}(\boldsymbol{\theta})$, by the light green polygon and the identified individual constitutive linear constraints by the dark green, claret red, purple, yellow, cyan and dark blue lines over the region formed by the boundary sets of feasible (blue) and infeasible (red) points. By comparing the identified constitutive linear constraints given by Equation (6.21) with the actual linear inequality constraints given in Equation (6.19), it can be concluded that our algorithm identifies the actual linear inequality constraints successfully. This example also proves that our algorithm successfully detects the actual bound constraints given in Equation (6.19a-d) by the use of constitutive linear inequality constraints. It is also possible to observe from Figure 6.9 that the aggregated constraint does not lead to any violations at the boundary and thus satisfies

all the feasible and infeasible boundary points. The zero values of $J_1$and J_2 imply this as well.

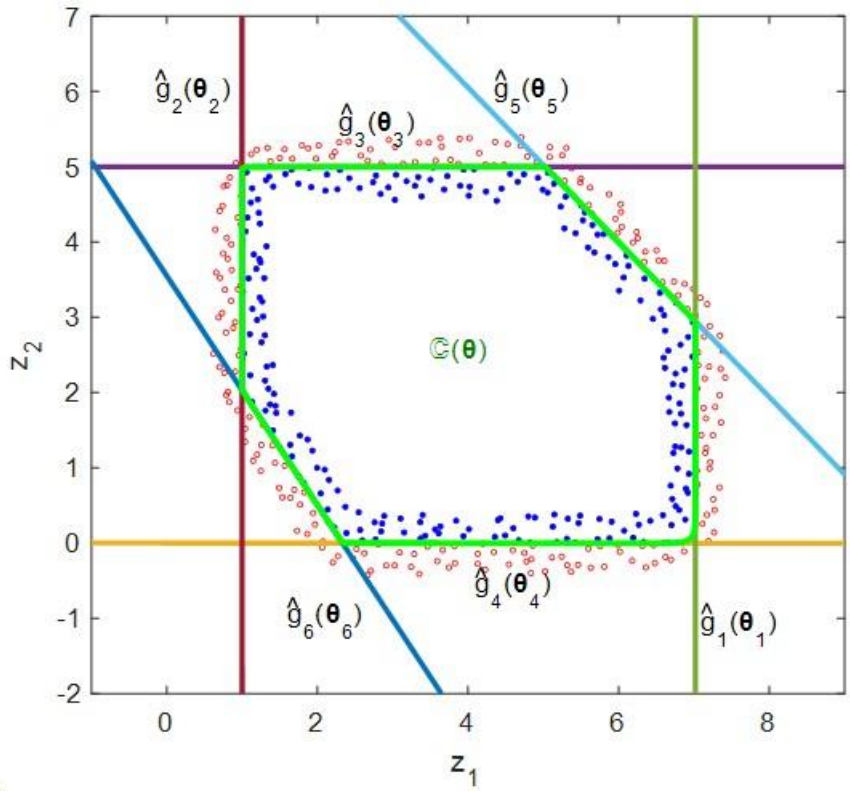

Figure 6.9. Identified optimal constitutive linear constraints and aggregated constraint superimposed with the boundary sets of feasible and infeasible points.

6.5. Identification of Mixture of Inequality Constraints

In this section, we will demonstrate the ability of our algorithm to identify mixture of two-dimensional inequality constraints forming a non-convex feasible region. These constraints are four bounds, one linear, one circular, and one parabolic constraints. Seven inequality constraints which will be treated as unknowns and will be detected by our algorithm are given by the following equation:

$$g_1: \quad +z_1 - 7 \leq 0, \tag{6.22a}$$

$$g_2: \quad -z_1 + 1 \leq 0, \tag{6.22b}$$

$$g_3: \quad +z_2 - 5 \leq 0, \tag{6.22c}$$

$$g_4: \quad -z_2 \leq 0, \tag{6.22d}$$

$$g_5: \quad +z_1 + z_2 - 10 \leq 0, \tag{6.22e}$$

$$g_6: \quad -(z_1 - 2)^2 - (z_2 - 1)^2 + 4 \leq 0, \tag{6.22f}$$

$$g_7: \quad -10(z_1 - 5)^2 + z_2 - 2 \leq 0. \tag{6.22g}$$

It should be also noted that the type of the inequality constraint given in Equation (6.22g) is parabolic which has not been introduced in Chapter 5. Our algorithm will utilize $\theta_2(z_1 - \theta_1)^2 - \theta_3 z_2 + \theta_0 \leq 0$ as the constitutive two-dimensional parabolic inequality constraint. $N^{Prbl}(D + 2)$ parameters are introduced for N^{Prbl} of such parabolic constraints. Additionally, since the outside of the circular constraint is set to be feasible, the constitutive circular constraint is multiplied by minus one.

SLHS is employed to generate the sample points. The cardinality of the set of sample points is $N = 5000$. The lower and upper bounds on the sample points are specified as $\mathbf{z}^{LB} = [-1 \ -2]$ and $\mathbf{z}^{UB} = [+9 \ +7]$. The number of constitutive inequality constraints is $N^{Con} = 7$ including $N^{Bnds} = 2D = 4$, $N^{Lin} = 1$, $N^{Circ} = 1$ and $N^{Prbl} = 1$. The value of ρ is specified as 500. The complete set of points, $\mathbf{Z}$, is a 5041×2 matrix. The sets of feasible, $\mathbf{X}$, and infeasible, $\mathbf{Y}$, points, generated by the sampling procedure (mentioned in Chapter 3), are 1000×2 and 4041×2 matrices respectively, i.e., $N^f = 1000$ and $N^i = 4041$. In this example, p^{th} percentile is specified as $p = 1$. Thus, the boundary sets of feasible, $\mathbf{X}^b$, and infeasible, $\mathbf{Y}^b$, points become 431×2 and 453×2 matrices respectively, i.e., $N^{fb} = 431$ and $N^{ib} = 453$.

The number of parameters (optimization decision variables) becomes $|\boldsymbol{\theta}| = 14$. The values of lower bounds are set to −10 and the values of upper bounds are set to +10 for each of the decision variables and the initial population size is taken as $NP = 40$. Our algorithm utilizes all parts of the overall objective function introduced in Chapter 5. The value of the weighting parameter, μ, of third sub-objective function, J_3, is set to 10^{-7}.

The optimization terminates at the function tolerance of 10^{-9} after 2951 iterations with 100505 function evaluations and gives the minimum value of the overall objective function as $J = 21.22 \times 10^{-7}$. The values of the first and second parts of the objective

function become zero. i.e., $J_1 = 0$ and $J_2 = 0$, while the value of the third part of the objective function becomes $J_3 = 21.22$.

The constitutive inequality constraints identified are given explicitly together with the optimal values of parameters as:

$$\hat{g}_1(\theta_1) = +z_1 - 6.992, \tag{6.23a}$$

$$\hat{g}_2(\theta_2) = -z_1 + 0.989, \tag{6.23b}$$

$$\hat{g}_3(\theta_3) = +z_2 - 4.990, \tag{6.23c}$$

$$\hat{g}_4(\theta_4) = -z_2 - 0.000, \tag{6.23d}$$

$$\hat{g}_5(\boldsymbol{\theta}_5) = +0.008z_1 + 0.008z_2 - 0.080, \tag{6.23e}$$

$$\hat{g}_6(\boldsymbol{\theta}_6) = -(z_1 - 1.965)^2 - (z_2 - 0.971)^2 + 4.121, \tag{6.23f}$$

$$\hat{g}_7(\boldsymbol{\theta}_7) = -0.163(z_1 - 4.997)^2 + 0.016z_2 - 0.032. \tag{6.23g}$$

Additionally, the identified linear and parabolic constraints normalized by the related parameters are shown as:

$$\frac{\hat{g}_5(\boldsymbol{\theta}_5)}{0.008} = +1.000z_1 + 1.000z_2 - 10.000, \tag{6.24a}$$

$$\frac{\hat{g}_7(\boldsymbol{\theta}_7)}{0.016} = -10.030(z_1 - 4.997)^2 + 1.000z_2 - 2.009. \tag{6.24b}$$

Figure 6.10 presents the aggregated constraint, $\mathbb{C}(\boldsymbol{\theta})$, by the light green and the identified individual constitutive constraints by the yellow, purple, dark green, cyan, claret red lines, dark blue circle, and orange parabola over the region formed by the boundary sets of feasible (blue) and infeasible (red) points. By comparing the identified constitutive constraints given by Equation (6.23) and Equation (6.24) with the actual inequality constraints given in Equation (6.22), it can be concluded that our algorithm identifies the actual linear inequality constraints successfully. It is also possible to observe from Figure 6.10 that the aggregated constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible boundary points. The zero values of $J_1$and J_2 imply this as well.

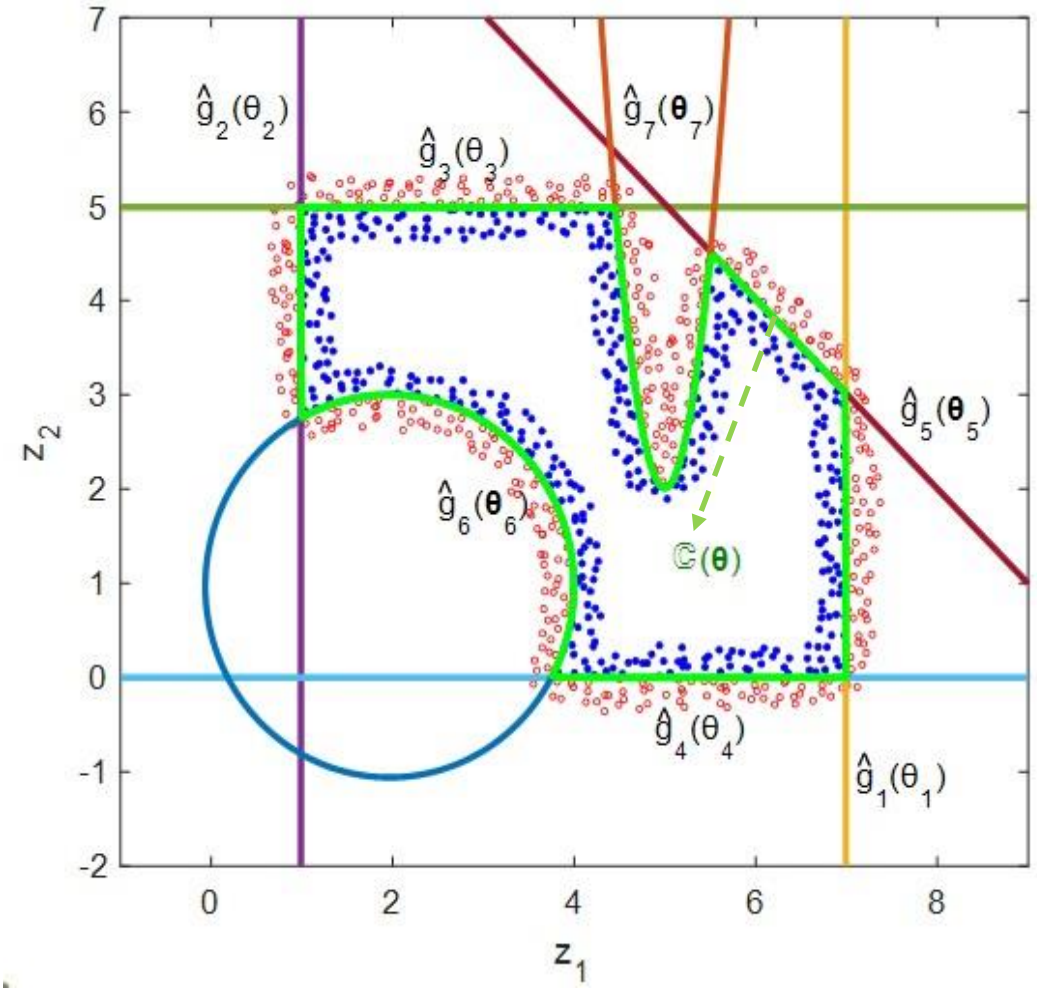

Figure 6.10. Identified optimal constitutive constraints and aggregated constraint superimposed with the boundary sets of feasible and infeasible points.

7. GENERALIZED CONSTRAINT IDENTIFICATION VIA DESIGN MATRIX

In this chapter, we will equip our algorithm with the design-matrix approach so as to build the functional-form-free constitutive constraints and to rebuild some of the form-specific constitutive constraints to identify the unknown inequality constraints forming the boundary between the feasible and infeasible regions. This chapter begins with the introduction to some of the design matrices utilized in this chapter together with the short explanation of other possibilities of the design-matrix concept. Afterwards, we will form the constitutive inequality constraints by the design-matrix approach and compare the form-free constitutive constraints, which will be introduced in this chapter for the first time, with some of the form-specific constitutive constraints that were introduced in Chapter 5. Lastly, by fabricating a toy scenario of a mixture of constitutive constraints, we will demonstrate the way our algorithm sorts and aggregates the constitutive constraints into an augmented design matrix.

The design matrix, also known as the model matrix or factor matrix, is commonly used to form a multiple-regression models (Khattree and Naik, 2000; Johnson and Wichern, 2007; Montgomery and Runger, 2018). However, in this thesis work, we employ the design matrix to create form-free constitutive inequality constraints. The design-matrix approach allows us not only to extend the form-specific constitutive constraints to form-free ones but also to embed many other types of form-specific constitutive constraints into the design matrix. Thus, the use of the design matrix provides tremendous design and computational flexibility in modelling and identification of the constitutive inequality constraints.

Each column of any design matrix is a factor (term) contributing to the constitutive inequality constraints. In other words, the columns are the building blocks (factors) of the form-free or form-specific constitutive inequality constraints. Each row of the design matrix corresponds to each sample point. In this thesis work, we will employ the symbol $\mathbf{\Phi}$ to denote the design matrix. To build the constitutive inequality constraints, the design matrices are created by utilizing the options of MATLAB's "x2fx" function --even though

not obligatory-- except for the constitutive bound constraints. MATLAB's "x2fx" function is limited to "linear", "interaction", "pure quadratic", and "quadratic" models in its straightforward implementation, however it is possible to use advanced options of the "x2fx" function to form almost infinite variety of models as long as the models involve powers of the factors. It is of course possible to form design matrix directly without using "x2fx" function which provides even more flexibility such as exponentiation of the factors, factors that involve trigonometric functions, ratios, etc.

With the "linear" option of MATLAB's "x2fx" function, we can create a design matrix to build the form-specific constitutive linear inequality constraint. This type model matrix is explicitly given as:

$$\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z}) = \begin{bmatrix} 1 & z_1^1 & z_2^1 & \cdots & z_D^1 \\ 1 & z_1^2 & z_2^2 & \cdots & z_D^2 \\ 1 & z_1^3 & z_2^3 & \cdots & z_D^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & z_1^N & z_2^N & \cdots & z_D^N \end{bmatrix}, \tag{7.1a}$$

$$\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z}) = \begin{bmatrix} \mathbf{1} & \mathbf{z}_1 & \mathbf{z}_2 & \cdots & \mathbf{z}_D \end{bmatrix}, \tag{7.1b}$$

where $\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z}) \in \mathbb{R}^{N\times(D+1)}$ includes a building block (column) of constant term and the building blocks (columns) of the linear terms for each of dimensions. For the set of feasible points, $\mathbf{Z}$ is replaced with $\mathbf{X}$, and the factor matrix becomes $\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{X}) \in \mathbb{R}^{N^f\times(D+1)}$. For the set of infeasible points, $\mathbf{Z}$ is replaced with $\mathbf{Y}$, and the factor matrix becomes $\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Y}) \in \mathbb{R}^{N^i\times(D+1)}$. Each row of $\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})$ corresponds to each sample (feasible or infeasible) generated as explained in Chapter 3.

With the "pure quadratic" option of MATLAB's "x2fx" function, we can create a design matrix to build the form-free second-order (pure quadratic) constitutive constraint. Here, different from the "linear" option we say "form-free" since the "pure quadratic" function embeds linear terms as well as squared powers. Thus, with the "pure quadratic" we can identify a linear constraint (at the end of optimization the coefficients of the power terms become zero) or a pure circular constraint or any other combination remaining effective after optimization. This type of model matrix is explicitly given as:

$$\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z}) = \begin{bmatrix} 1 & \mathrm{z}_1^1 & \mathrm{z}_2^1 & \cdots & \mathrm{z}_{\mathrm{D}}^1 & (\mathrm{z}_1^1)^2 & (\mathrm{z}_2^1)^2 & \cdots & (\mathrm{z}_{\mathrm{D}}^1)^2 \\ 1 & \mathrm{z}_1^2 & \mathrm{z}_2^2 & \cdots & \mathrm{z}_{\mathrm{D}}^2 & (\mathrm{z}_1^2)^2 & (\mathrm{z}_2^2)^2 & \cdots & (\mathrm{z}_{\mathrm{D}}^2)^2 \\ 1 & \mathrm{z}_1^3 & \mathrm{z}_2^3 & \cdots & \mathrm{z}_{\mathrm{D}}^3 & (\mathrm{z}_1^3)^2 & (\mathrm{z}_2^3)^2 & \cdots & (\mathrm{z}_{\mathrm{D}}^3)^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \mathrm{z}_1^{\mathrm{N}} & \mathrm{z}_2^{\mathrm{N}} & \cdots & \mathrm{z}_{\mathrm{D}}^{\mathrm{N}} & (\mathrm{z}_1^{\mathrm{N}})^2 & \left(\mathrm{z}_2^{\mathrm{N}}\right)^2 & \cdots & \left(\mathrm{z}_{\mathrm{D}}^{\mathrm{N}}\right)^2 \end{bmatrix}, \tag{7.2a}$$

$$\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z}) = \begin{bmatrix} \boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z}) & \vdots & \mathbf{z}_1^2 & \mathbf{z}_2^2 & \cdots & \mathbf{z}_{\mathrm{D}}^2 \end{bmatrix}, \tag{7.2b}$$

where $\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z}) \in \mathbb{R}^{\mathrm{N}\times(2\mathrm{D}+1)}$ includes a building block of constant term, the building blocks of linear, and squared terms for each dimension. For the set of feasible points, $\mathbf{Z}$ is replaced with $\mathbf{X}$, and the factor matrix becomes $\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{X}) \in \mathbb{R}^{\mathrm{N}^{\mathrm{f}}\times(2\mathrm{D}+1)}$. For the set of infeasible points, $\mathbf{Z}$ is replaced with $\mathbf{Y}$, and the factor matrix becomes $\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Y}) \in \mathbb{R}^{\mathrm{N}^{\mathrm{i}}\times(2\mathrm{D}+1)}$. As can be seen in Equation (7.2), such a design matrix is nothing but the combination of the design matrices belonging to the form-specific constitutive linear constraint and the elementwise squared terms for each of dimension.

With the "interaction" option of MATLAB's "x2fx" function, we can create a design matrix, to build the form-free constitutive constraint (linear or nonlinear), containing factors' linear-interaction terms (columns). This type model matrix is explicitly given as:

$$\boldsymbol{\Phi}^{\mathrm{I}}(\mathbf{Z}) = \begin{bmatrix} \boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z}) & \vdots & \mathbf{z}_1\mathbf{z}_2 & \cdots & \mathbf{z}_1\mathbf{z}_{\mathrm{D}} & \mathbf{z}_2\mathbf{z}_3 & \cdots & \mathbf{z}_2\mathbf{z}_{\mathrm{D}} & \cdots & \mathbf{z}_{\mathrm{D}-1}\mathbf{z}_{\mathrm{D}} \end{bmatrix}, \tag{7.3}$$

where $\boldsymbol{\Phi}^{\mathrm{I}}(\mathbf{Z}) \in \mathbb{R}^{\mathrm{N}\times(\mathrm{D}+\binom{\mathrm{D}}{2}+1)}$ consists of a building block of the constant term, the building blocks of linear terms for each dimension, and the building blocks of the interaction terms. For the set of feasible points, $\mathbf{Z}$ is replaced with $\mathbf{X}$, and the factor matrix becomes $\boldsymbol{\Phi}^{\mathrm{I}}(\mathbf{X}) \in \mathbb{R}^{\mathrm{N}^{\mathrm{f}}\times(\mathrm{D}+\binom{\mathrm{D}}{2}+1)}$. For the set of infeasible points, $\mathbf{Z}$ is replaced with $\mathbf{Y}$, and the factor matrix becomes $\boldsymbol{\Phi}^{\mathrm{I}}(\mathbf{Y}) \in \mathbb{R}^{\mathrm{N}^{\mathrm{i}}\times(\mathrm{D}+\binom{\mathrm{D}}{2}+1)}$. As can be observed from Equation (7.3), such a design matrix is the combination of the linear design matrix and the columns for the elementwise interaction terms.

With the "quadratic" option of MATLAB's "x2fx" function, we can create a design matrix to build a form-free constitutive quadratic constraint. This model matrix is explicitly given as:

$$\boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{Z}) = \begin{bmatrix} \boldsymbol{\Phi}^{\mathrm{I}}(\mathbf{Z}) & \vdots & \mathbf{z}_1^2 & \mathbf{z}_2^2 & \cdots & \mathbf{z}_{\mathrm{D}}^2 \end{bmatrix}, \tag{7.4}$$

where $\boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{Z}) \in \mathbb{R}^{\mathrm{N}\times(2\mathrm{D}+\binom{\mathrm{D}}{2}+1)}$ consists of a building block of the constant term, the building block of the linear terms for each dimension, the building block of the interaction

terms, and the building block of the squared terms for each dimension. For the set of feasible points, $\mathbf{Z}$ is replaced with $\mathbf{X}$, and the factor matrix becomes $\boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{X}) \in \mathbb{R}^{\mathrm{N^f}\times(2\mathrm{D}+\binom{\mathrm{D}}{2}+1)}$. For the set of infeasible points, $\mathbf{Z}$ is replaced with $\mathbf{Y}$, and the factor matrix becomes $\boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{Y}) \in \mathbb{R}^{\mathrm{N^i}\times(2\mathrm{D}+\binom{\mathrm{D}}{2}+1)}$. As can be observed from Equation (7.4), such a design matrix is the combination of the design matrix including factors' linear-interaction terms (columns) and the elementwise squared terms for each dimension.

In order to provide our algorithm with the ability to utilize the form-specific constitutive bound constraints together with the constitutive constraints which are built through the design matrix approach, we first create a design matrix for the bound constraints. This type of the model matrix is explicitly given as:

$$\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_\mathrm{i}) = \quad [\mathbf{1} \quad +\mathbf{z}_\mathrm{i}]; \;\; \mathrm{i} = 1, \dots, \mathrm{D}, \tag{7.5a}$$

$$\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_\mathrm{i}) = \quad [\mathbf{1} \quad -\mathbf{z}_\mathrm{i}]; \;\; \mathrm{i} = 1, \dots, \mathrm{D}, \tag{7.5b}$$

where $\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_\mathrm{i}) \in \mathbb{R}^{\mathrm{N}\times 2}$ includes a building block of the constant term, sticking with a building block for the positive sign of the linear terms of the i^{th} dimension, $\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_\mathrm{i}) \in \mathbb{R}^{\mathrm{N}\times 2}$ consists of a building block of the constant term, sticking with a building block for the negative sign of the linear terms of the i^{th} dimension. For the feasible points, $\mathbf{z}_\mathrm{i}$ is replaced with $\mathbf{x}_\mathrm{i}$, and the factor matrices become $\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{x}_\mathrm{i}) \in \mathbb{R}^{\mathrm{N^f}\times 2}$ and $\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{x}_\mathrm{i}) \in \mathbb{R}^{\mathrm{N^f}\times 2}$ For the infeasible points, $\mathbf{z}_\mathrm{i}$ is replaced with $\mathbf{y}_\mathrm{i}$, and the factor matrices become $\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{y}_\mathrm{i}) \in \mathbb{R}^{\mathrm{N^i}\times 2}$ and $\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{y}_\mathrm{i}) \in \mathbb{R}^{\mathrm{N^i}\times 2}$.

The establishment of the form-free constitutive constraints can be assisted with the design matrices beyond the ones introduced above. For instance, the building blocks (columns) of the model matrix may be formed by employing sine or cosine functions or by utilizing an exponential function. Additionally, the order of the constitutive constraints which will be supported by the factor matrices introduced via MATLAB's "x2fx" function will become two at most. This may not suffice to approximate and thus to identify some inequality constraints forming an intricately-shaped feasible regions. Thus, we can create the design matrix to build the form-free constitutive constraints with term orders higher than second-order to resolve this issue. Actually, we will shed light into this topic by the

deployment of some of the machine-learning algorithms rather than creating large factor matrices with too many columns of higher order.

After the above introduction to some types of design matrices, we can begin building some specific constitutive constraints.

It should be highlighted that all rows of $\boldsymbol{\Phi}(\mathbf{X})$, multiplied by $\boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Type}}$ which is the optimal values of the parameters of related constitutive constraint (column coefficients of $\boldsymbol{\Phi}$), satisfy all the individual constitutive inequality constraints, i.e., $\hat{\mathrm{g}}_{\mathrm{j}}\left(\mathbf{x}^{l}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Type}}\right) \coloneqq \left[\boldsymbol{\phi}(\mathbf{x}^{l})\boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Type}}\right] \leq 0\,,\ \forall\, l \in \left\{1, \dots, \mathrm{N}^{\mathrm{f}}\right\},\ \forall\, \mathrm{n} \in \{1, \dots, \mathrm{N}^{\mathrm{Type}}\},\ \forall\, \mathrm{j} \in \{1, \dots, \mathrm{N}^{\mathrm{Con}}\}$. However, some rows of $\boldsymbol{\Phi}(\mathbf{Y})$, multiplied by $\boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Type}}$ which is the optimal values of the parameters of related constitutive constraint (column coefficients of $\boldsymbol{\Phi}$), may satisfy some -but not all- individual constitutive inequality constraints, i.e., $\hat{\mathrm{g}}_{\mathrm{j}}\left(\mathbf{y}^{k}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Type}}\right) \coloneqq \left[\boldsymbol{\phi}(\mathbf{y}^{k})\boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Type}}\right] < 0,\ \exists\, k \in \left\{1, \dots, \mathrm{N}^{\mathrm{i}}\right\}, \exists\, \mathrm{n} \in \{1, \dots, \mathrm{N}^{\mathrm{Type}}\},\ \exists\, \mathrm{j} \in \{1, \dots, \mathrm{N}^{\mathrm{Con}}\}$. Therefore, when $\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Y}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Type}}\right) > \mathbf{0}$ is written below, it implies that this is true at least for one row of $\boldsymbol{\Phi}(\mathbf{Y})$ multiplied by $\boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Type}}$, but may not be true for all rows of $\boldsymbol{\Phi}(\mathbf{Y})$ multiplied by $\boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Type}}$.

The form-specific constitutive bound constraints can be reformulated via the design matrix approach and can be represented with respect to each dimension of each feasible point as $\boldsymbol{\phi}^{\mathrm{UB}}(\mathrm{x}_{\mathrm{i}}^{l})\boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{UB}} \leq 0$ and $\boldsymbol{\phi}^{\mathrm{LB}}(\mathrm{x}_{\mathrm{i}}^{l})\boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{LB}} \leq 0$ and with respect to some dimension of some infeasible points $\boldsymbol{\phi}^{\mathrm{UB}}(\mathrm{y}_{\mathrm{i}}^{k})\boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{UB}} > 0$ and $\boldsymbol{\phi}^{\mathrm{LB}}(\mathrm{y}_{\mathrm{i}}^{k})\boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{LB}} > 0$, where $\boldsymbol{\phi}^{\mathrm{UB}}(\mathrm{x}_{\mathrm{i}}^{l}) \in \mathbb{R}^{1\times 2}$ and $\boldsymbol{\phi}^{\mathrm{LB}}(\mathrm{x}_{\mathrm{i}}^{l}) \in \mathbb{R}^{1\times 2}$ denote each row of $\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{x}_{\mathrm{i}})$ and $\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{x}_{\mathrm{i}})$ respectively for $l = \left\{1, \dots, \mathrm{N}^{\mathrm{f}}\right\}$ and $\mathrm{i} = \{1, \dots, \mathrm{D}\}$, $\boldsymbol{\phi}^{\mathrm{UB}}(\mathrm{y}_{\mathrm{i}}^{k}) \in \mathbb{R}^{1\times 2}$ and $\boldsymbol{\Phi}^{\mathrm{LB}}(\mathrm{y}_{\mathrm{i}}^{k}) \in \mathbb{R}^{1\times 2}$ denote some row of $\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{y}_{\mathrm{i}})$ and $\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{y}_{\mathrm{i}})$ respectively, and $\boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{UB}} = \left[\theta_{\mathrm{i}}^{\mathrm{U}}, 1\right]^{\mathrm{T}} \in \mathbb{R}^{2}$ and $\boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{LB}} = \left[\theta_{\mathrm{i}}^{\mathrm{L}}, 1\right]^{\mathrm{T}} \in \mathbb{R}^{2}$ includes the parameters of the constitutive upper- and lower-bound constraints respectively. The number of the constitutive bound constraints becomes $\mathrm{N}^{\mathrm{B}} = 2\mathrm{D}$ and thus $2\mathrm{D}$ parameters are added to the optimization problem for N^{B} such constitutive constraints.

These constitutive constraints, when written in the generic forms $\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{UB}}\right)$ and $\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{LB}}\right)$, take the form given as:

$$\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{UB}}\right) \coloneqq \quad \boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_{\mathrm{i}})\boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{UB}}; \qquad \begin{cases} \mathrm{j} = 1, \dots, \mathrm{D} \\ \mathrm{i} = 1, \dots, \mathrm{D} \end{cases}, \tag{7.6a}$$

$$\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{LB}}\right) \coloneqq \quad \boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_{\mathrm{i}})\boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{LB}}; \qquad \begin{cases} \mathrm{j} = 1 + \mathrm{D}, \dots, 2\mathrm{D} \\ \mathrm{i} = 1, \dots, \mathrm{D} \end{cases}. \tag{7.6b}$$

For the feasible points, $\mathbf{Z}$ is replaced with $\mathbf{X}$, and the constitutive constraints are treated as $\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{X}, \boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{UB}}\right) \leq \mathbf{0}$ and $\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{X}, \boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{LB}}\right) \leq \mathbf{0}$. For the infeasible points, $\mathbf{Z}$ is replaced with $\mathbf{Y}$, and the direction of the inequalities for the infeasible points must be changed and thus the constitutive constraints for infeasible points are treated as $\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Y}, \boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{UB}}\right) > \mathbf{0}$ and $\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Y}, \boldsymbol{\theta}_{\mathrm{i}}^{\mathrm{LB}}\right) > \mathbf{0}$.

The form-specific constitutive linear inequality constraints can be reformulated via the design matrix approach with respect to each feasible point as $\boldsymbol{\phi}^{\mathrm{L}}(\mathbf{x}^{l})\boldsymbol{\theta}^{\mathrm{L}} \leq 0$ and with respect to each infeasible point as $\boldsymbol{\phi}^{\mathrm{L}}(\mathbf{y}^{k})\boldsymbol{\theta}^{\mathrm{L}} > 0$, where $\boldsymbol{\phi}^{\mathrm{L}}(\mathbf{x}^{l}) \in \mathbb{R}^{1\times(\mathrm{D}+1)}$ and $\boldsymbol{\phi}^{\mathrm{L}}(\mathbf{y}^{k}) \in \mathbb{R}^{1\times(\mathrm{D}+1)}$ denote each row of $\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{X})$ and $\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Y})$, respectively, $\boldsymbol{\theta}^{\mathrm{L}} = \left[\theta_0^{\mathrm{Lin}}, \dots, \theta_{\mathrm{D}}^{\mathrm{Lin}}\right]^{\mathrm{T}} \in \mathbb{R}^{\mathrm{D}+1}$ consists of the parameters of the constitutive linear constraint. Thus, $\mathrm{D}+1$ parameters are required per a such constitutive constraint. Accordingly, $\mathrm{N}^{\mathrm{L}}(\mathrm{D}+1)$ parameters are added to the optimization problem when our algorithm utilizes N^{L} such constitutive constraints.

These constitutive constraints, when written in the generic form $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{L}})$, take the form given as:

$$\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{L}}) \coloneqq \quad \boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})\boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{L}}; \qquad \begin{cases} \mathrm{j} = 1, \dots, \mathrm{N}^{\mathrm{L}} \\ \mathrm{n} = 1, \dots, \mathrm{N}^{\mathrm{L}} \end{cases}. \tag{7.7}$$

For the set of feasible points, $\mathbf{Z}$ is replaced with $\mathbf{X}$, and the constitutive constraint is treated as $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{X}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{L}}) \leq \mathbf{0}$. For the set of infeasible points, $\mathbf{Z}$ is replaced with $\mathbf{Y}$, and the direction of the inequalities of infeasible points must be changed and thus the constitutive constraint for infeasible points is treated as $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Y}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{L}}) > \mathbf{0}$.

The form-free constitutive linear-interaction constraints can be formulated via the design matrix approach and can be represented with respect to each feasible point as

$\boldsymbol{\phi}^{\mathrm{I}}(\mathbf{x}^l)\boldsymbol{\theta}^{\mathrm{I}} \leq 0$ and with respect to each infeasible point as $\boldsymbol{\phi}^{\mathrm{I}}(\mathbf{y}^k)\boldsymbol{\theta}^{\mathrm{I}} > 0$, where $\boldsymbol{\phi}^{\mathrm{I}}(\mathbf{x}^l) \in \mathbb{R}^{1\times(\mathrm{D}+\binom{\mathrm{D}}{2}+1)}$ and $\boldsymbol{\phi}^{\mathrm{I}}(\mathbf{y}^k) \in \mathbb{R}^{1\times(\mathrm{D}+\binom{\mathrm{D}}{2}+1)}$ imply each row of $\boldsymbol{\Phi}^{\mathrm{I}}(\mathbf{X})$ and $\boldsymbol{\Phi}^{\mathrm{I}}(\mathbf{Y})$, respectively, and $\boldsymbol{\theta}^{\mathrm{I}} = \left[\theta_0^{\mathrm{I}}, \dots, \theta_{\mathrm{D}+\binom{\mathrm{D}}{2}}^{\mathrm{I}}\right]^{\mathrm{T}} \in \mathbb{R}^{\mathrm{D}+\binom{\mathrm{D}}{2}+1}$ includes the parameters of this type of the form-free constitutive constraint. Thus, $\mathrm{D} + \binom{\mathrm{D}}{2} + 1$ parameters are required per a such constitutive constraint. Accordingly, $\mathrm{N}^{\mathrm{I}}\left(\mathrm{D} + \binom{\mathrm{D}}{2} + 1\right)$ parameters are added to the optimization problem when our algorithm utilizes N^{I} such constitutive constraints.

These constitutive constraints, when written in the generic form $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{I}})$, take the form given as:

$$\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{I}}) \coloneqq \quad \boldsymbol{\Phi}^{\mathrm{I}}(\mathbf{Z})\boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{I}}; \qquad \begin{cases} \mathrm{j} = 1, \dots, \mathrm{N}^{\mathrm{I}} \\ \mathrm{n} = 1, \dots, \mathrm{N}^{\mathrm{I}} \end{cases}. \tag{7.8}$$

For the set of feasible points, $\mathbf{Z}$ is replaced with $\mathbf{X}$, and the constitutive constraint is treated as $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{X}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{I}}) \leq \mathbf{0}$. For the set of infeasible points, $\mathbf{Z}$ is replaced with $\mathbf{Y}$, and the direction of the inequalities of infeasible points must be changed and thus the constitutive constraint for infeasible points is treated as $\hat{\mathbf{g}}_{\mathrm{j}}(\mathbf{Y}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{I}}) > \mathbf{0}$.

The form-free constitutive pure quadratic constraints can be formed via the design matrix approach and can be represented with respect to each feasible point as $\boldsymbol{\phi}^{\mathrm{PQ}}(\mathbf{x}^l)\boldsymbol{\theta}^{\mathrm{PQ}} \leq 0$ and with respect to each infeasible points as $\boldsymbol{\phi}^{\mathrm{PQ}}(\mathbf{y}^k)\boldsymbol{\theta}^{\mathrm{PQ}} > 0$, where $\boldsymbol{\phi}^{\mathrm{PQ}}(\mathbf{x}^l) \in \mathbb{R}^{1\times(2\mathrm{D}+1)}$ and $\boldsymbol{\phi}^{\mathrm{PQ}}(\mathbf{y}^k) \in \mathbb{R}^{1\times(2\mathrm{D}+1)}$ denote each row of $\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{X})$ and $\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Y})$, respectively, and $\boldsymbol{\theta}^{\mathrm{PQ}} = \left[\theta_0^{\mathrm{PQ}}, \dots, \theta_{2\mathrm{D}}^{\mathrm{PQ}}\right]^{\mathrm{T}} \in \mathbb{R}^{2\mathrm{D}+1}$ includes the parameters of this type of the form-free constitutive constraint. Thus, $2\mathrm{D} + 1$ parameters are required per a such constitutive constraint. Accordingly, $\mathrm{N}^{\mathrm{PQ}}(2\mathrm{D} + 1)$ parameters are added to the optimization problem when our algorithm utilizes N^{PQ} such constitutive constraints.

These constitutive constraints, when written in the generic form $\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{PQ}}\right)$, take the form given as:

$$\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{PQ}}\right) \coloneqq \quad \boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})\boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{PQ}}; \qquad \begin{cases} \mathrm{j} = 1, \dots, \mathrm{N}^{\mathrm{PQ}} \\ \mathrm{n} = 1, \dots, \mathrm{N}^{\mathrm{PQ}} \end{cases}. \tag{7.9}$$

For the set of feasible points, $\mathbf{Z}$ is replaced with $\mathbf{X}$, and the constitutive constraint is treated as $\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{X}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{PQ}}\right) \leq \mathbf{0}$. For the set of infeasible points, $\mathbf{Z}$ is replaced with $\mathbf{Y}$, and the direction of the inequalities of infeasible points must be changed and thus the constitutive constraint for infeasible points is treated as $\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Y}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{PQ}}\right) > \mathbf{0}$.

The form-free constitutive quadratic constraints can be formulated via the design matrix approach and can be represented with respect to each feasible point as $\boldsymbol{\phi}^{\mathrm{Q}}(\mathbf{x}^{l})\boldsymbol{\theta}^{\mathrm{Q}} \leq 0$ and with respect to each infeasible point as $\boldsymbol{\phi}^{\mathrm{Q}}(\mathbf{y}^{k})\boldsymbol{\theta}^{\mathrm{Q}} > 0$, where $\boldsymbol{\phi}^{\mathrm{Q}}(\mathbf{x}^{l}) \in \mathbb{R}^{1\times\left(2\mathrm{D}+\binom{\mathrm{D}}{2}+1\right)}$ and $\boldsymbol{\phi}^{\mathrm{Q}}(\mathbf{y}^{k}) \in \mathbb{R}^{1\times\left(2\mathrm{D}+\binom{\mathrm{D}}{2}+1\right)}$ imply each row of $\boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{X})$ and $\boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{Y})$ respectively, and $\boldsymbol{\theta}^{\mathrm{Q}} = \left[\theta_0^{\mathrm{Q}}, \dots, \theta_{\left(2\mathrm{D}+\binom{\mathrm{D}}{2}\right)}^{\mathrm{Q}}\right]^{\mathrm{T}} \in \mathbb{R}^{2\mathrm{D}+\binom{\mathrm{D}}{2}+1}$ includes the parameters of this type of the form-free constitutive constraint. Thus, $2\mathrm{D} + \binom{\mathrm{D}}{2} + 1$ parameters are required per a such constitutive constraint. Accordingly, $\mathrm{N}^{\mathrm{Q}}\left(2\mathrm{D} + \binom{\mathrm{D}}{2} + 1\right)$ parameters are added to the optimization problem when our algorithm utilizes N^{Q} such constitutive constraints.

These constitutive constraints, when written in the generic form $\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Q}}\right)$, take the form given as:

$$\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Q}}\right) \coloneqq \quad \boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{Z})\boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Q}}; \qquad \begin{cases} \mathrm{j} = 1, \dots, \mathrm{N}^{\mathrm{Q}} \\ \mathrm{n} = 1, \dots, \mathrm{N}^{\mathrm{Q}} \end{cases}. \tag{7.10}$$

For the set of feasible points, $\mathbf{Z}$ is replaced with $\mathbf{X}$, and the constitutive constraint is treated as $\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{X}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Q}}\right) \leq \mathbf{0}$. For the set of infeasible points, $\mathbf{Z}$ is replaced with $\mathbf{Y}$, the direction of the inequalities of infeasible points must be changed and thus the constitutive constraint for infeasible points is treated as $\hat{\mathbf{g}}_{\mathrm{j}}\left(\mathbf{Y}, \boldsymbol{\theta}_{\mathrm{n}}^{\mathrm{Q}}\right) > \mathbf{0}$.

It should also be highlighted that the set of feasible and infeasible points, $\mathbf{X}$ and $\mathbf{Y}$, can directly be replaced with the boundary set of feasible and infeasible points, $\mathbf{X}^{\mathrm{b}}$ and $\mathbf{Y}^{\mathrm{b}}$, in the above paragraphs if the reduced sets are utilized in constraint identification. In such cases, N, N^{f}, and N^{i} must also be replaced with N^{b}, N^{fb}, and N^{ib}, respectively.

These constitutive constraints that can be obtained via design matrix are explicitly given in Table 7.1 (explicit forms of the constitutive ellipsoidal and circular constraints had been introduced in Chapter 5).

Table 7.1. Explicit mathematical representations of constitutive inequality constraints that can be obtained via design matrix approach for the two-dimensional case.

Type of Constitutive Inequality Constraints	**Parametric Form of Constitutive Inequality Constraint** $\hat{g}(z_1, z_2, \boldsymbol{\theta})$	**Eq. No**
Bounds	$\theta_1^U + z_1 \leq 0,$ $\theta_1^L - z_1 \leq 0,$ $\theta_2^U + z_2 \leq 0,$ $\theta_2^L - z_2 \leq 0,$	(7.11a)
Linear	$\theta_0^{Lin} + \theta_1^{Lin} z_1 + \theta_2^{Lin} z_2 \leq 0,$	(7.11b)
Linear-Interaction	$\theta_0^I + \theta_1^I z_1 + \theta_2^I z_2 + \theta_3^I z_1 z_2 \leq 0,$	(7.11c)
Pure Quadratic	$\theta_0^{PQ} + \theta_1^{PQ} z_1 + \theta_2^{PQ} z_2 + \theta_3^{PQ} z_1^2 + \theta_4^{PQ} z_2^2 \leq 0,$	(7.11d)
Quadratic	$\theta_0^Q + \theta_1^Q z_1 + \theta_2^Q z_2 + \theta_3^Q z_1 z_2 + \theta_4^Q z_1^2 + \theta_5^Q z_2^2 \leq 0,$	(7.11e)
Circular	$\theta_3 + \theta_4 z_1 + \theta_5 z_2 + z_1^2 + z_2^2 \leq 0,$	(7.11f)
Ellipsoidal	$\theta_6 + \theta_7 z_1 + \theta_8 z_2 + \theta_9 z_1 z_2 + \theta_{10} z_1^2 + \theta_{11} z_2^2 \leq 0.$	(7.11g)

By looking at Equation (7.11b-e), it is possible to observe that the form-free constitutive quadratic constraint includes the form-specific constitutive linear and the form-free constitutive linear-interaction and pure quadratic constraints. This means that our algorithm can find all unknown inequality constraints which can be approximated with the constitutive constraints given in Equation (7.11b-e) by the form-free quadratic constraint. In other words, linear, pure quadratic, and linear-interaction terms are all embedded in the full-quadratic design matrix. That is, if at the end of optimization $\theta_3^Q = \theta_4^Q = \theta_5^Q = 0$, then linear constraint is obtained, and if $\theta_4^Q = \theta_5^Q = 0$, linear-interaction is obtained, and if $\theta_3^Q = 0$, pure quadratic is obtained. Additionally, the form-free constitutive pure quadratic

and linear-interaction constraints also includes all terms of the constitutive linear constraint. In other words, the simplest constitutive constraint is linear, while the most developed constitutive constraint is quadratic in terms of their capabilities to detect different types of the unknown inequality constraints. Lastly, the constitutive ellipsoidal and circular constraints (see Table 5.1) are given in a way that the terms included in the squared terms of the constitutive circular constraint and the squared and interaction terms of the constitutive ellipsoidal constraint are written explicitly in Equation (7.11e-f). By such representation for the constitutive circular and ellipsoidal constraints, it is possible to see that the form-free quadratic constraint best suits for the constitutive circular constraint, and thus its deployment is more appropriate for approximating the unknown circular constraints. Furthermore, the form-free quadratic constitutive constraint is the same as the explicit form of the constitutive ellipsoidal constraint given in Equation (7.11g), and hence its usage is convenient for approximating unknown ellipsoidal inequality constraints. However, for instance, as it can be deduced, it is impossible to approximate an unknown ellipsoidal inequality constraint by the use of form-free constitutive pure quadratic constraint.

Lastly, we can illustrate how our design-matrix algorithm handles mixture of constitutive constraints employing a toy scenario. For instance, given that the number of the constitutive constraints utilized by our algorithm is $\mathrm{N}^{\mathrm{Con}} = 4$ (four constitutive constraints in total), including $\mathrm{N}^{\mathrm{L}} = 1$, $\mathrm{N}^{\mathrm{I}} = 1$, $\mathrm{N}^{\mathrm{PQ}} = 1$, $\mathrm{N}^{\mathrm{Q}} = 1$, i.e., one constitutive linear, one constitutive linear-interaction, one constitutive pure quadratic, and one constitutive full-quadratic inequality constraints, our algorithm combines their design matrices and their parameters into separate matrices and evaluates the values of the constitutive constraints over N points as given in:

$$\begin{bmatrix} \hat{\mathbf{g}}_1^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}^{\mathrm{L}}) \\ \dots\dots\dots\dots\dots \\ \hat{\mathbf{g}}_2^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}^{\mathrm{I}}) \\ \dots\dots\dots\dots\dots \\ \hat{\mathbf{g}}_3^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}^{\mathrm{PQ}}) \\ \dots\dots\dots\dots\dots \\ \hat{\mathbf{g}}_4^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}^{\mathrm{Q}}) \end{bmatrix}^{\mathrm{T}} \coloneqq \boldsymbol{\Phi}(\mathbf{Z})\mathbf{P} = \left(\begin{bmatrix} (\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})\boldsymbol{\theta}^{\mathrm{L}})^{\mathrm{T}} \\ \dots\dots\dots\dots\dots \\ (\boldsymbol{\Phi}^{\mathrm{I}}(\mathbf{Z})\boldsymbol{\theta}^{\mathrm{I}})^{\mathrm{T}} \\ \dots\dots\dots\dots\dots \\ (\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})\boldsymbol{\theta}^{\mathrm{PQ}})^{\mathrm{T}} \\ \dots\dots\dots\dots\dots \\ (\boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{Z})\boldsymbol{\theta}^{\mathrm{Q}})^{\mathrm{T}} \end{bmatrix}_{\mathrm{N}^{\mathrm{Con}}\times\mathrm{N}} \right)^{\mathrm{T}}, \tag{7.12a}$$

$$\boldsymbol{\Phi}(\mathbf{Z}) = \left[\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z}) \ \vdots \ \boldsymbol{\Phi}^{\mathrm{I}}(\mathbf{Z}) \ \vdots \ \boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z}) \ \vdots \ \boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{Z})\right]_{\mathrm{N}\times\left(6\mathrm{D}+2\binom{\mathrm{D}}{2}+4\right)}, \tag{7.12b}$$

$$\mathbf{P} = \begin{bmatrix} \boldsymbol{\theta}^{\mathrm{L}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \boldsymbol{\theta}^{\mathrm{I}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}^{\mathrm{PQ}} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}^{\mathrm{Q}} \end{bmatrix}_{(6\mathrm{D}+2\binom{\mathrm{D}}{2}+4)\times \mathrm{N}^{\mathrm{Con}}} . \tag{7.12c}$$

$\boldsymbol{\Phi}(\mathbf{Z}) \in \mathbb{R}^{\mathrm{N}\times(6\mathrm{D}+2\binom{\mathrm{D}}{2}+4)}$ is the augmented design matrix including the design matrices of the constitutive constraints, $\mathbf{P} \in \mathbb{R}^{(6\mathrm{D}+2\binom{\mathrm{D}}{2}+4)\times \mathrm{N}^{\mathrm{Con}}}$ is the parameters matrix (which is a sparse matrix) whose columns respectively comprise the parameters of the constitutive linear, $\boldsymbol{\theta}^{\mathrm{L}} \in \mathbb{R}^{\mathrm{D}+1}$, linear-interaction, $\boldsymbol{\theta}^{\mathrm{I}} \in \mathbb{R}^{\mathrm{D}+\binom{\mathrm{D}}{2}+1}$, pure-quadratic, $\boldsymbol{\theta}^{\mathrm{PQ}} \in \mathbb{R}^{2\mathrm{D}+1}$, and quadratic, $\boldsymbol{\theta}^{\mathrm{Q}} \in \mathbb{R}^{2\mathrm{D}+\binom{\mathrm{D}}{2}+1}$, constraints. The row cardinality of the transpose of the parameters matrix is the column cardinality, which is $|\mathbf{P}^{\mathrm{T}}| = \mathrm{N}^{\mathrm{Con}}$. $\boldsymbol{\Phi}(\mathbf{Z})$ and $\mathbf{P}$ are explicitly shown in Equation (7.12b) and Equation (7.12c) only for the scenario mentioned above.

If our algorithm utilizes the constitutive bound constraints together with the constitutive constraints provided through the scenario given above, the number of the constitutive constraints becomes $\mathrm{N}^{\mathrm{Con}} = 2\mathrm{D} + 4$ including $\mathrm{N}^{\mathrm{B}} = 2\mathrm{D}$, $\mathrm{N}^{\mathrm{L}} = 1$, $\mathrm{N}^{\mathrm{I}} = 1$, $\mathrm{N}^{\mathrm{PQ}} = 1$, $\mathrm{N}^{\mathrm{Q}} = 1$. The values of the constitutive constraints are evaluated over N points as given in:

$$\begin{bmatrix} \hat{\mathbf{g}}_1^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_1^{\mathrm{UB}}) \\ \cdots\cdots\cdots \\ \vdots \\ \cdots\cdots\cdots \\ \hat{\mathbf{g}}_{\mathrm{D}}^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{D}}^{\mathrm{UB}}) \\ \cdots\cdots\cdots \\ \hat{\mathbf{g}}_{\mathrm{D}+1}^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_1^{\mathrm{LB}}) \\ \cdots\cdots\cdots \\ \vdots \\ \cdots\cdots\cdots \\ \hat{\mathbf{g}}_{2\mathrm{D}}^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_{\mathrm{D}}^{\mathrm{LB}}) \\ \cdots\cdots\cdots \\ \hat{\mathbf{g}}_{2\mathrm{D}+1}^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}^{\mathrm{L}}) \\ \cdots\cdots\cdots \\ \hat{\mathbf{g}}_{2\mathrm{D}+2}^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}^{\mathrm{I}}) \\ \cdots\cdots\cdots \\ \hat{\mathbf{g}}_{2\mathrm{D}+3}^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}^{\mathrm{PQ}}) \\ \cdots\cdots\cdots \\ \hat{\mathbf{g}}_{2\mathrm{D}+4}^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}^{\mathrm{Q}}) \end{bmatrix}^{\mathrm{T}} \coloneqq \boldsymbol{\Phi}(\mathbf{Z})\mathbf{P} = \left(\begin{bmatrix} \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_1)\boldsymbol{\theta}_1^{\mathrm{UB}}\right)^{\mathrm{T}} \\ \cdots\cdots\cdots \\ \vdots \\ \cdots\cdots\cdots \\ \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_{\mathrm{D}})\boldsymbol{\theta}_{\mathrm{D}}^{\mathrm{UB}}\right)^{\mathrm{T}} \\ \cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_1)\boldsymbol{\theta}_1^{\mathrm{LB}})^{\mathrm{T}} \\ \cdots\cdots\cdots \\ \vdots \\ \cdots\cdots\cdots \\ \left(\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_{\mathrm{D}})\boldsymbol{\theta}_{\mathrm{D}}^{\mathrm{LB}}\right)^{\mathrm{T}} \\ \cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})\boldsymbol{\theta}^{\mathrm{L}})^{\mathrm{T}} \\ \cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{I}}(\mathbf{Z})\boldsymbol{\theta}^{\mathrm{I}})^{\mathrm{T}} \\ \cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})\boldsymbol{\theta}^{\mathrm{PQ}})^{\mathrm{T}} \\ \cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{Z})\boldsymbol{\theta}^{\mathrm{Q}})^{\mathrm{T}} \end{bmatrix}_{\mathrm{N}^{\mathrm{Con}}\times \mathrm{N}} \right)^{\mathrm{T}} . \tag{7.13}$$

$\boldsymbol{\Phi}(\mathbf{Z}) \in \mathbb{R}^{N\times\left(10D+2\binom{D}{2}+4\right)}$ is the augmented design matrix including the design matrices of the constitutive constraints, $\mathbf{P} \in \mathbb{R}^{\left(10D+2\binom{D}{2}+4\right)\times N^{Con}}$ is the sparse parameters matrix, the columns of which include the parameters of only one constitutive constraint. As can be understood by Equation (7.12) and Equation (7.13), our algorithm sorts the constitutive constraints in a way that it starts with the N^{B} constitutive bound constraints and continue with the N^{L} constitutive linear, N^{I} constitutive linear-interaction, N^{PQ} constitutive pure quadratic constraints. It completes sorting of the constitutive constraints with the N^{Q} constitutive quadratic constraints.

Our algorithm aggregates one- (Chapter 5) or multiple-type (Chapter 7) of constitutive constraints by taking advantage of the KS aggregation function. The aggregation of constitutive constraints are given by:

$$\mathbb{C}(\mathbf{Z},\mathbf{P}) \coloneqq \overline{KS}_{U}\,(\mathbf{Z},\mathbf{P},\rho), \tag{7.14a}$$

$$\mathbb{C}(\mathbf{X},\mathbf{P}) \coloneqq \max_{j}\left\{\boldsymbol{\Phi}(\mathbf{X})\mathbf{p}_{j}\right\} + \frac{1}{\rho}\ln\left(\sum_{j=1}^{N^{Con}} e^{\rho\left(\boldsymbol{\Phi}(\mathbf{X})\mathbf{p}_{j}-\max_{j}\left\{\boldsymbol{\Phi}(\mathbf{X})\mathbf{p}_{j}\right\}\right)}\right) \leq \mathbf{0}, \tag{7.14b}$$

$$\mathbb{C}(\mathbf{Y},\mathbf{P}) \coloneqq \max_{j}\left\{\boldsymbol{\Phi}(\mathbf{Y})\mathbf{p}_{j}\right\} + \frac{1}{\rho}\ln\left(\sum_{j=1}^{N^{Con}} e^{\rho\left(\boldsymbol{\Phi}(\mathbf{Y})\mathbf{p}_{j}-\max_{j}\left\{\boldsymbol{\Phi}(\mathbf{Y})\mathbf{p}_{j}\right\}\right)}\right) > \mathbf{0}. \tag{7.14c}$$

The aggregation of the N^{Con} (or, $|\mathbf{P}^{T}|$) constitutive constraints can be shown as in Equation (7.14a) where $\mathbf{p}_{j} \in \mathbb{R}^{|\mathbf{P}|}$ comprises the parameters of the j^{th} constitutive constraint and the zero values corresponding to the elements of the design matrices of the other constitutive constraints, and implies that the j^{th} column of $\mathbf{P}$, $\mathbb{C}(\mathbf{Z},\mathbf{P}) \in \mathbb{R}^{N}$ includes the values of the KS-aggregated constraint evaluated at all sample points. For the set of feasible points, $\mathbf{Z}$ is replaced with $\mathbf{X}$, and the values of the aggregated constraint evaluated at the feasible points should become $\mathbb{C}(\mathbf{X},\mathbf{P}) \leq \mathbf{0} \in \mathbb{R}^{N^{f}}$ upon successful termination of the optimization yielding the optimal values of parameters, $\mathbf{P}$, which is explicitly illustrated in Equation (7.14b). For the set of infeasible points, $\mathbf{Z}$ is replaced with $\mathbf{Y}$, and the values of the aggregated constraint evaluated at the infeasible points should become $\mathbb{C}(\mathbf{Y},\mathbf{P}) > \mathbf{0} \in \mathbb{R}^{N^{i}}$ with the optimal parameters, which is explicitly shown in Equation (7.14c). In the nutshell, N^{Con} constitutive constraints are reduced to the one aggregated constraint

evaluated over N^f feasible and N^i infeasible points, and hence, over N points, by the KS aggregation function. This topic of aggregation of the constraints via the KS function had been thoroughly discussed in Chapter 4.

After obtaining $\mathbb{C}(\mathbf{X}, \mathbf{P})$ and $\mathbb{C}(\mathbf{Y}, \mathbf{P})$ by the aggregation of the constitutive constraints built via the design matrix approach, our algorithm employs the overall objective function (see Chapter 5) given as:

$$\min_{\boldsymbol{\theta}} \; J(\mathbf{Z}, \mathbf{P}) = J_1(\mathbf{Z}, \mathbf{P}) + J_2(\mathbf{Z}, \mathbf{P}) + \mu J_3(\mathbf{P}), \tag{7.15}$$

to reach the optimal values of the parameters. The parts of the overall objective function and the optimization procedure had been thoroughly discussed in Chapter 5.

At the onset of this chapter, we had mentioned that MATLAB's "x2fx" function was limited to "linear", "interaction", "pure quadratic", and "quadratic" models in its straightforward implementation, and that it was possible to use advanced options of the "x2fx" function to form almost infinite variety of models as long as the models involve powers of the factors. We had also mentioned that it was also possible to form the design matrix directly without using "x2fx" function to provide even more flexibility by forming the columns (factors) of the design matrix with arbitrary functions such as exponential, trigonometric, ratio, etc. At the closure of this chapter, we present a generic design matrix (independent of MATLAB's x2fx function) with the following equation:

$$\boldsymbol{\Phi}^G(\mathbf{Z}) = [\boldsymbol{\Phi}(\mathbf{Z}) \;\vdots\; f_1(\mathbf{Z}) \;\vdots\; f_2(\mathbf{Z}) \;\vdots\; \cdots \;\vdots\; f_m(\mathbf{Z})]. \tag{7.16}$$

In this generic design matrix, the $\boldsymbol{\Phi}(\mathbf{Z})$ related columns may be the columns of the standard linear, linear interaction, pure quadratic, quadratic functions (one or all of them in column sequel) if they are included in the generic design matrix. The arbitrary function columns $f_1(\mathbf{Z}), \ldots, f_m(\mathbf{Z})$ may be any arbitrary function of $\mathbf{Z}$, for instance, $f_1(\mathbf{Z}) = e^{\mathbf{Z}}$, $f_2(\mathbf{Z}) = \sin(\mathbf{Z}) + \cos(\mathbf{Z})$, $f_3(\mathbf{Z}) = 1/(1 + \mathbf{Z})$, $f_4(\mathbf{Z}) = \sin(\mathbf{Z})/\cos(\mathbf{Z})$, $f_5(\mathbf{Z}) = J_0(\mathbf{Z})$ $f_6(\mathbf{Z}) = 1/(1 + e^{-\mathbf{Z}})$, $f_7(\mathbf{Z}) = e^{\mathbf{Z}} \tanh(\mathbf{Z})$, etc. Here, it should be understood that the nonlinear functions apply to all or selected elements (coordinates) of $\mathbf{Z}$, e.g., $f_3(\mathbf{Z})$ may act on only $\mathbf{z}_1$ component as $1/(1 + \mathbf{z}_1)$ whereas $f_1(\mathbf{Z})$ may act on all dimensions as $[e^{\mathbf{z}_1} \; e^{\mathbf{z}_2} \cdots e^{\mathbf{z}_D} \; e^{\mathbf{z}_1} e^{\mathbf{z}_2} \cdots e^{\mathbf{z}_1} e^{\mathbf{z}_D} \cdots]$. In other words, each $f(\mathbf{Z})$ may be either a column vector or a sub matrix in Equation (7.16). Of course, $f(\mathbf{Z})$'s may represent the power series

of the individual elements or any linear or nonlinear combination of all elements in $\mathbf{Z}$, e.g., as $[\mathbf{Z}^1\ \mathbf{Z}^2\ \mathbf{Z}^3 \cdots\ \mathbf{Z}^{\mathrm{m}}]$. Another possibility is that the $\mathrm{f}(\mathbf{Z})$'s may be elements of $(\mathrm{m}+1)^{\mathrm{D}}$ factorial design of coordinates, where m is the model order (number of design levels) and D is the number of factors (number of coordinates). The parameters (decision variables in optimization) are the column coefficients of this generic design matrix. With our novelty of using design matrix in constraint-identification task, we opened infinite number of possibilities and thus tremendous flexibility in approximating unknown constraints forming inexpressible feasible- and infeasible-region boundaries. This idea of the use of generic design matrix in constraint-identification task has led us to further propose the use of machine learning tools with the aim to completely get rid of functional-form dependence in constraint-identification task and this will be the topic of Chapter 9.

8. CONSTRAINT IDENTIFICATION EXAMPLES SOLVED VIA DESIGN MATRIX

In this chapter, we will present constraint-identification examples detected by our algorithm equipped with the design matrix. With the first example for 2-D case, we will demonstrate the ability of the form-specific constitutive linear constraints and the form-free constitutive pure quadratic constraints (two scenarios in one example) to detect a rectangular feasible region. Afterwards, we will present three constraint-identification examples for 2-D case and one constraint-identification example for 3-D case. Lastly, we will complete this chapter with the constraint-identification examples which will demonstrate the ability of some of the form-free constitutive constraints deployed with the generalized design matrices that were briefly introduced at the closure of Chapter 7. We will also pave the road further towards the idea of constraint-identification via form-free, implicitly nonlinear approximators of $\mathbb{C}(\boldsymbol{\theta})$ that can be obtained by the deployment of machine learning tools.

8.1. The First Example for 2-D Case

In this section, we will demonstrate the ability of our algorithm to detect the rectangular feasible region formed by unknown bound constraints via design matrix approach. Four bound constraints which will be treated as unknowns and will be detected by our algorithm are given by the following equation:

$$g_1: \quad +z_1 - 7 \leq 0, \tag{8.1a}$$

$$g_2: \quad -z_1 + 1 \leq 0, \tag{8.1b}$$

$$g_3: \quad +z_2 - 5 \leq 0, \tag{8.1c}$$

$$g_4: \quad -z_2 \leq 0. \tag{8.1d}$$

For this example, we will represent two scenarios to detect the rectangular feasible region, and thus to identify unknown bound constraints. One of them includes the deployment of the four form-specific constitutive linear constraints and the other is the

deployment of the two form-free constitutive pure quadratic constraints to demonstrate that it is possible to identify unknown bound constraints by different types and different number of constitutive constraints.

For both scenarios, SLHS (Chapter 3) is employed to generate sample points. For the first scenario, information on the sampling and optimization, as well as the results are all presented in Table 8.1.

The values of perimeter and area of the detected feasible region (computed via two third-party utility functions, "plotcontour" and "interpclosed", that can be found in MATLAB's File Exchange site) are also presented in Table 8.1.

Table 8.1. Sampling and optimization information and the results.

Sampling Information				
$\mathbf{z}^{\mathrm{LB}} = [-1 \ -2]$	$\mathbf{z}^{\mathrm{UB}} = [+9 \ +7]$			
$\mathrm{N} = 5000$	$\mathrm{N}^{\mathrm{f}} = 1682$			$\mathrm{N}^{\mathrm{i}} = 3359$
$\mathbf{Z} \in \mathbb{R}^{5000\times 2}$	$\mathbf{X} \in \mathbb{R}^{1682\times 2}$			$\mathbf{Y} \in \mathbb{R}^{3359\times 2}$
	$\mathrm{p} = 1$			
$\mathrm{N}^{\mathrm{b}} = 1009$	$\mathrm{N}^{\mathrm{fb}} = 480$			$\mathrm{N}^{\mathrm{ib}} = 529$
$\mathbf{Z} \in \mathbb{R}^{1009\times 2}$	$\mathbf{X}^{\mathrm{b}} \in \mathbb{R}^{480\times 2}$			$\mathbf{Y}^{\mathrm{b}} \in \mathbb{R}^{529\times 2}$
Optimization Settings				
$\mathrm{N}^{\mathrm{Con}} = \mathrm{N}^{\mathrm{L}} = 4$	$\mathrm{D} = 2$	$\lvert\boldsymbol{\theta}\rvert = 12$	$\mu = 10^{-6}$	$\rho = 500$
$\mathbf{z}^{\mathrm{L}} = [-10 \ -10]$	$\mathbf{z}^{\mathrm{U}} = [+10 \ +10]$		$\mathrm{NP} = 40$	$\mathrm{Tol} = 10^{-9}$
Optimization Results				
$\mathrm{Iterations} = 16483$	$\mathrm{Function\ Evaluations} = 591494$			
$\mathrm{J} = 3.351 \times 10^{-7}$	$\mathrm{J}_1 = 0$	$\mathrm{J}_2 = 0$	$\mathrm{J}_3 = 0.335$	
$\mathrm{Perimeter} = 21.611$	$\mathrm{Area} = 30.011$			

The form-specific constitutive linear inequality constraints identified and the optimal values of parameters are all given by:

$$\begin{bmatrix} \hat{\mathbf{g}}_1^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}_1^{\mathrm{L}}) \\ \cdots\cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_2^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}_2^{\mathrm{L}}) \\ \cdots\cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_3^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}_3^{\mathrm{L}}) \\ \cdots\cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_4^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}_4^{\mathrm{L}}) \end{bmatrix}^{\mathrm{T}} \coloneqq \boldsymbol{\Phi}(\mathbf{Z})\mathbf{P} = \left(\begin{bmatrix} (\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})\boldsymbol{\theta}_1^{\mathrm{L}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})\boldsymbol{\theta}_2^{\mathrm{L}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})\boldsymbol{\theta}_3^{\mathrm{L}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})\boldsymbol{\theta}_4^{\mathrm{L}})^{\mathrm{T}} \end{bmatrix}_{4\times 1009} \right)^{\mathrm{T}}, \tag{8.2a}$$

$$\boldsymbol{\Phi}(\mathbf{Z}) = [\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z}) \vdots \quad \boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z}) \vdots \quad \boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z}) \vdots \quad \boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})]_{1009\times 12}, \tag{8.2b}$$

$$\mathbf{P} = \begin{bmatrix} \boldsymbol{\theta}_1^{\mathrm{L}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \boldsymbol{\theta}_2^{\mathrm{L}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}_3^{\mathrm{L}} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}_4^{\mathrm{L}} \end{bmatrix}_{12\times 4}, \tag{8.2c}$$

$$\boldsymbol{\theta}_1^{\mathrm{L}} = \begin{bmatrix} +3.946 \times 10^{-8} \\ -1.095 \times 10^{-7} \\ -4.323 \times 10^{-2} \end{bmatrix}_{3\times 1}, \tag{8.2d}$$

$$\boldsymbol{\theta}_2^{\mathrm{L}} = \begin{bmatrix} +1.363 \times 10^{-2} \\ -1.364 \times 10^{-2} \\ -2.525 \times 10^{-6} \end{bmatrix}_{3\times 1}, \tag{8.2e}$$

$$\boldsymbol{\theta}_3^{\mathrm{L}} = \begin{bmatrix} -9.617 \times 10^{-2} \\ -1.178 \times 10^{-4} \\ +1.930 \times 10^{-2} \end{bmatrix}_{3\times 1}, \tag{8.2f}$$

$$\boldsymbol{\theta}_4^{\mathrm{L}} = \begin{bmatrix} -1.302 \times 10^{-1} \\ +1.863 \times 10^{-2} \\ -1.270 \times 10^{-4} \end{bmatrix}_{3\times 1}, \tag{8.2g}$$

where Equation (8.2a) shows implicit constraints in matrix form, as introduced in Chapter 7, Equation (8.2b) exhibits the augmented design matrix formed by the design sub-matrices belonging to the form-free constitutive linear constraints (as in Equation (7.1)). Equation (8.2c) displays the parameters matrix created when the four constitutive linear constraints are employed.

Figure 8.1 presents the aggregated constraint, $\mathbb{C}(\mathbf{P})$, by the light green rectangular curve and the identified individual constitutive linear constraints by the claret red, cyan, purple, and yellow lines over the region formed by the boundary sets of feasible (blue) and infeasible (red) points.

By just comparing the identified constitutive linear constraints given by Equation (8.2) with the actual linear inequality constraints given in Equation (8.1), it can be difficult

to realize the success of our algorithm since the values of parameters of identified constitutive constraints are different than those of the unknown inequality constraints due to the non-uniqueness of the coefficients of inequality constraints (see Chapter 5). However, it is possible to observe from Figure 8.1 that the aggregated constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible boundary points. The zero values of J_1 and J_2 in Table 8.1 imply this as well.

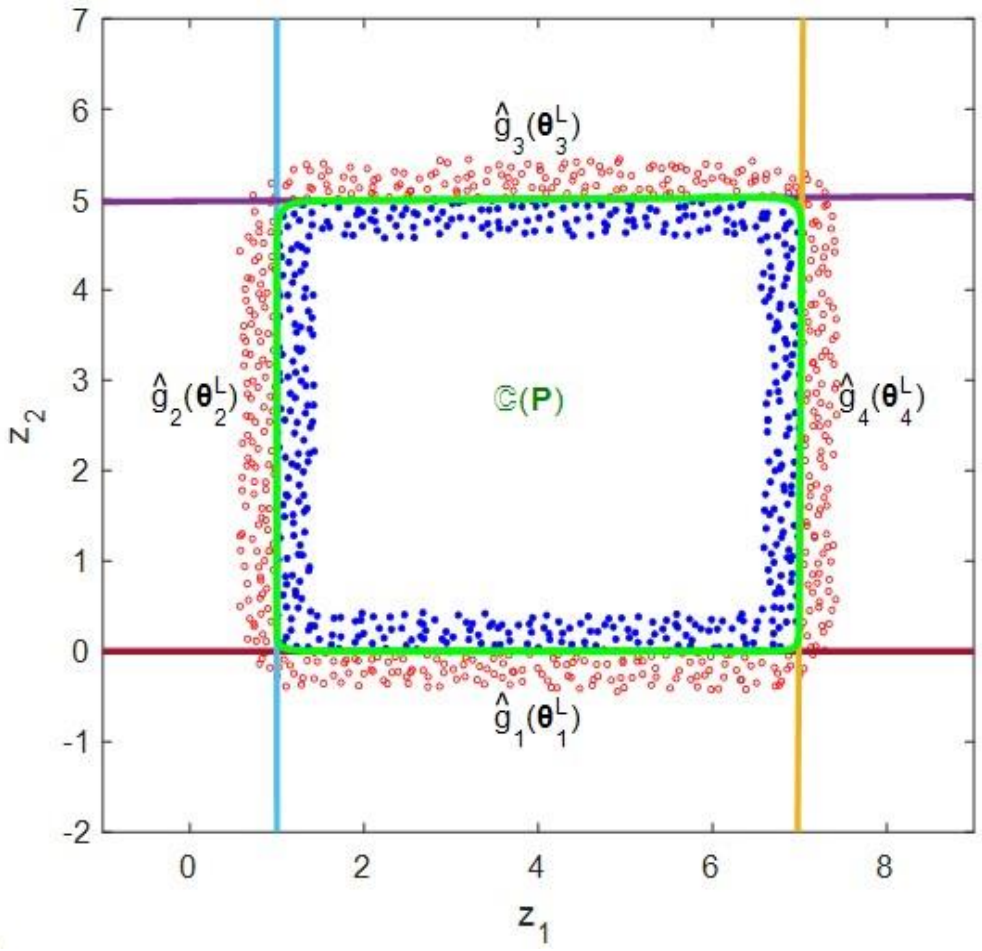

Figure 8.1. Identified optimal constitutive linear constraints and aggregated constraint superimposed with the boundary sets of feasible and infeasible points.

When it comes to the second scenario, information on the sampling and optimization, as well as the results are all presented in Table 8.2.

The form-free constitutive pure quadratic inequality constraints identified and the optimal values of parameters are all given by:

$$\begin{bmatrix} \hat{\mathbf{g}}_1^{\mathrm{T}}\left(\mathbf{Z}, \boldsymbol{\theta}_1^{\mathrm{PQ}}\right) \\ \cdots\cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_2^{\mathrm{T}}\left(\mathbf{Z}, \boldsymbol{\theta}_2^{\mathrm{PQ}}\right) \end{bmatrix}^{\mathrm{T}} \coloneqq \boldsymbol{\Phi}(\mathbf{Z})\mathbf{P} = \left(\begin{bmatrix} \left(\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})\boldsymbol{\theta}_1^{\mathrm{PQ}}\right)^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots\cdots \\ \left(\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})\boldsymbol{\theta}_2^{\mathrm{PQ}}\right)^{\mathrm{T}} \end{bmatrix}_{2\times 1012} \right)^{\mathrm{T}}, \tag{8.3a}$$

$$\boldsymbol{\Phi}(\mathbf{Z}) = [\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z}) \vdots \quad \boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})]_{1012\times 10}, \tag{8.3b}$$

$$\mathbf{P} = \begin{bmatrix} \boldsymbol{\theta}_1^{\mathrm{PQ}} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \boldsymbol{\theta}_2^{\mathrm{PQ}} \end{bmatrix}_{10\times 2}, \tag{8.3c}$$

$$\boldsymbol{\theta}_1^{\mathrm{PQ}} = \begin{bmatrix} +3.866\times 10^{-2} \\ -4.442\times 10^{-2} \\ +3.448\times 10^{-4} \\ +5.548\times 10^{-3} \\ -5.758\times 10^{-5} \end{bmatrix}_{5\times 1}, \tag{8.3d}$$

$$\boldsymbol{\theta}_1^{\mathrm{PQ}} = \begin{bmatrix} +2.397\times 10^{-4} \\ -1.991\times 10^{-5} \\ -3.936\times 10^{-2} \\ -3.180\times 10^{-6} \\ +7.863\times 10^{-3} \end{bmatrix}_{5\times 1}, \tag{8.3e}$$

where Equation (8.3a) shows implicit constraints in matrix form, as introduced in Chapter 7, Equation (8.3b) exhibits the augmented design matrix formed by the design sub-matrices belonging to the form-free constitutive pure quadratic constraints (as in Equation (7.2)). Equation (8.2c) displays the parameters matrix created by augmenting the parameters of the two constitutive pure quadratic constraints.

Figure 8.2 presents the aggregated constraint, $\mathbb{C}(\mathbf{P})$, by the light green rectangle and the identified individual constitutive pure quadratic constraints by the claret red, cyan lines over the region formed by the boundary sets of feasible (blue) and infeasible (red) points. By just comparing the identified constitutive pure quadratic constraints given by Equation (8.3) with the actual linear inequality constraints given in Equation (8.1), it is impossible to judge the success of our algorithm. This is because of the deployment of a different type of constitutive constraints, i.e., two pure quadratic constrains are used to identify the four linear constraints.

However, it is possible to observe from Figure 8.2 that the aggregated constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible boundary points. The zero values of J_1 and J_2 in Table 8.2 imply this as well.

Table 8.2. Sampling and optimization information and the results.

Sampling Information				
$\mathbf{z}^{\mathrm{LB}} = [-1 \ -2]$	$\mathbf{z}^{\mathrm{UB}} = [+9 \ +7]$			
$\mathrm{N} = 5000$	$\mathrm{N}^{\mathrm{f}} = 1682$			$\mathrm{N}^{\mathrm{i}} = 3359$
$\mathbf{Z} \in \mathbb{R}^{5000\times 2}$	$\mathbf{X} \in \mathbb{R}^{1682\times 2}$			$\mathbf{Y} \in \mathbb{R}^{3359\times 2}$
	$\mathrm{p} = 1$			
$\mathrm{N}^{\mathrm{b}} = 1012$	$\mathrm{N}^{\mathrm{fb}} = 480$			$\mathrm{N}^{\mathrm{ib}} = 532$
$\mathbf{Z} \in \mathbb{R}^{1012\times 2}$	$\mathbf{X}^{\mathrm{b}} \in \mathbb{R}^{480\times 2}$			$\mathbf{Y}^{\mathrm{b}} \in \mathbb{R}^{532\times 2}$
Optimization Settings				
$\mathrm{N}^{\mathrm{Con}} = \mathrm{N}^{\mathrm{PQ}} = 2$	$\mathrm{D} = 2$	$\lvert\boldsymbol{\theta}\rvert = 10$	$\mu = 10^{-6}$	$\rho = 500$
$\mathbf{z}^{\mathrm{L}} = [-10 \ -10]$	$\mathbf{z}^{\mathrm{U}} = [+10 \ +10]$		$\mathrm{NP} = 40$	$\mathrm{Tol} = 10^{-9}$
Optimization Results				
$\mathrm{Iterations} = 11434$	$\mathrm{Function\ Evaluations} = 438808$			
$\mathrm{J} = 1.365 \times 10^{-7}$	$\mathrm{J}_1 = 0$	$\mathrm{J}_2 = 0$	$\mathrm{J}_3 = 0.136$	
$\mathrm{Perimeter} = 21.774$	$\mathrm{Area} = 29.970$			

These two scenarios imply that with the deployment of form-specific constitutive bound constraints (the first scenario implies) and the deployment of the two form-free quadratic constitutive constraints (the second scenario implies), it is possible to detect the same rectangular feasible region with almost equal success. In other words, it is possible to approximate the feasible region formed by the four bound constraints through aggregating the two form-free constitutive pure quadratic constraints whose orders are greater than those of the form-specific constitutive linear constraints.

Additionally, the last three sections of this chapter will also exemplify the detection of different feasible regions with a single form-free constitutive constraints (one implicit function), as mentioned shortly at the closure of Chapter 7, whose orders will be even greater than those of the two types (pure quadratic and quadratic) of form-free constitutive constraints.

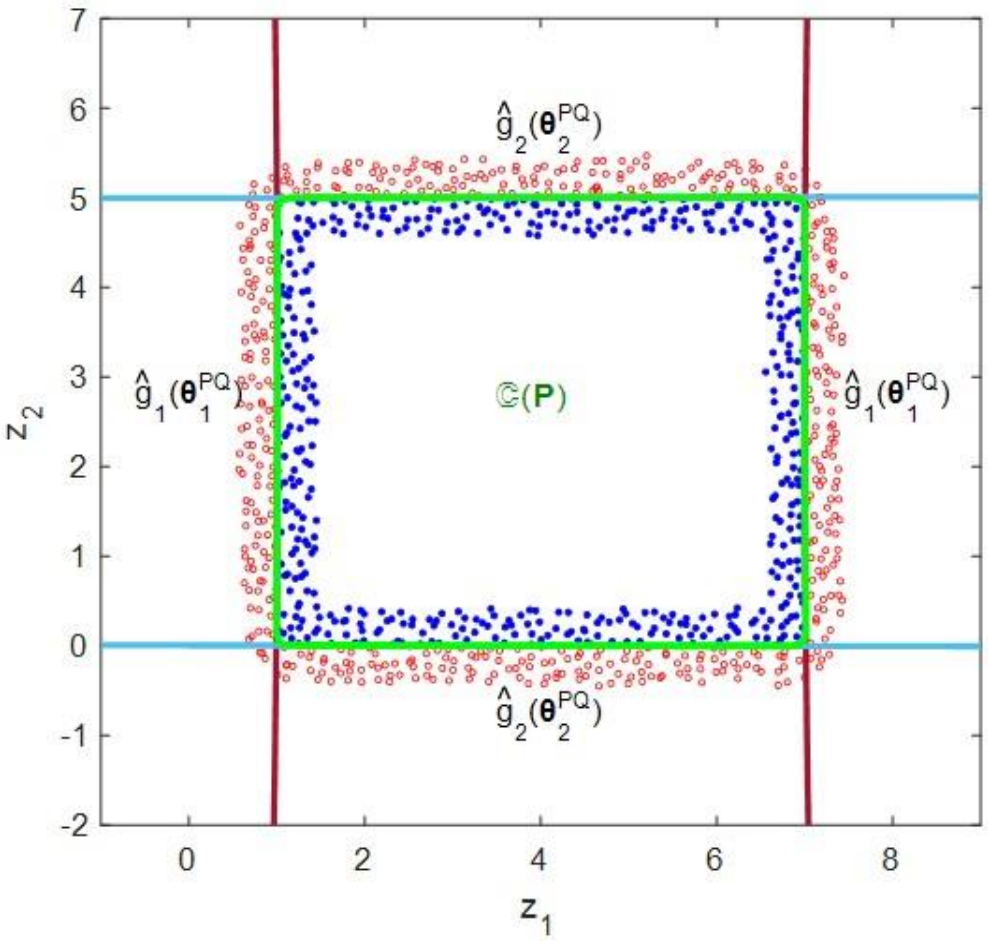

Figure 8.2. Identified optimal constitutive pure quadratic constraints and aggregated constraint superimposed with the boundary sets of feasible and infeasible points.

These examples have made us consider that the utilization of some machine-learning algorithm (to provide form-free implicit approximation) could be suitable for the constraint-identification task, and thus have paved the way towards the utilization of machine-learning algorithms such as Neural Networks and Extreme Learning Machine, even with single outputs, (as unspecified/undefined order, nonlinear, form-free implicit $\mathbb{C}(\boldsymbol{\theta})$ approximators). However, this is at the expense of sacrificing the extraction (identification) of explicit mathematical forms for the constitutive constraints.

8.2. The Second Example for 2-D Case

In this section, we will demonstrate the ability of our algorithm to detect the feasible region, which has been represented as the last example of Chapter 6 and detected by the form-specific constitutive constraints introduced in Chapter 5. Seven inequality constraints

which will be treated as unknowns and will be detected by our algorithm are given by the following equations:

$$g_1: \quad +z_1 - 7 \leq 0, \tag{8.4a}$$

$$g_2: \quad -z_1 + 1 \leq 0, \tag{8.4b}$$

$$g_3: \quad +z_2 - 5 \leq 0, \tag{8.4c}$$

$$g_4: \quad -z_2 \leq 0, \tag{8.4d}$$

$$g_5: \quad +z_1 + z_2 - 10 \leq 0, \tag{8.4e}$$

$$g_6: \quad -(z_1 - 2)^2 - (z_2 - 1)^2 + 4 \leq 0, \tag{8.4f}$$

$$g_7: \quad -10(z_1 - 5)^2 + z_2 - 2 \leq 0. \tag{8.4g}$$

SLHS is employed to generate sample points. For this example, information on the sampling and optimization, as well as the results are all presented in Table 8.3.

Table 8.3. Sampling and optimization information and the results.

Sampling Information			
$\mathbf{z}^{LB} = [-1 \ -2]$	$\mathbf{z}^{UB} = [+9 \ +7]$		
$N = 3500$	$N^f = 729$		$N^i = 2871$
$\mathbf{Z} \in \mathbb{R}^{3500\times2}$	$\mathbf{X} \in \mathbb{R}^{729\times2}$		$\mathbf{Y} \in \mathbb{R}^{2871\times2}$
	$p = 1$		
$N^b = 635$	$N^{fb} = 311$		$N^{ib} = 324$
$\mathbf{Z} \in \mathbb{R}^{635\times2}$	$\mathbf{X}^b \in \mathbb{R}^{311\times2}$		$\mathbf{Y}^b \in \mathbb{R}^{324\times2}$
Optimization Settings			
$N^B = 4$	$N^L = 1$	$N^{PQ} = 2$	
$N^{Con} = 7$	$D = 2$ $\quad \lvert\boldsymbol{\theta}\rvert = 17$	$\mu = 10^{-6}$	$\rho = 500$
$\mathbf{z}^L = [-10 \ -10]$	$\mathbf{z}^U = [+10 \ +10]$	$NP = 40$	$Tol = 10^{-9}$
Optimization Results			
Iterations $= 7874$	Function Evaluations $= 287298$		
$J = 2.420 \times 10^{-5}$	$J_1 = 0$	$J_2 = 0$	$J_3 = 24.209$
Perimeter $= 24.941$	Area $= 17.924$		

The constitutive inequality constraints identified and the optimal values of parameters are all given as:

$$\boldsymbol{\Phi}(\mathbf{Z})\mathbf{P} = \begin{bmatrix} \hat{\mathbf{g}}_1^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_1^{\mathrm{UB}}) \\ \cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_2^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_2^{\mathrm{UB}}) \\ \cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_3^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_1^{\mathrm{LB}}) \\ \cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_4^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_2^{\mathrm{LB}}) \\ \cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_5^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}^{\mathrm{L}}) \\ \cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_6^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_1^{\mathrm{PQ}}) \\ \cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_7^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_2^{\mathrm{PQ}}) \end{bmatrix}^{\mathrm{T}}$$

$$= \left(\begin{bmatrix} \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_1)\boldsymbol{\theta}_1^{\mathrm{UB}}\right)^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots \\ \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_2)\boldsymbol{\theta}_2^{\mathrm{UB}}\right)^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_1)\boldsymbol{\theta}_1^{\mathrm{LB}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_2)\boldsymbol{\theta}_2^{\mathrm{LB}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})\boldsymbol{\theta}^{\mathrm{L}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots \\ \left(\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})\boldsymbol{\theta}_1^{\mathrm{PQ}}\right)^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots \\ \left(\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})\boldsymbol{\theta}_2^{\mathrm{PQ}}\right)^{\mathrm{T}} \end{bmatrix}_{7\times 635} \right)^{\mathrm{T}}, \tag{8.5a}$$

$$\boldsymbol{\Phi}(\mathbf{Z}) = \left(\begin{bmatrix} \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_1)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_2)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_1)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_2)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})\right)^{\mathrm{T}} \end{bmatrix}_{21\times 635} \right)^{\mathrm{T}}, \tag{8.5b}$$

$$\mathbf{P} = \begin{bmatrix} \boldsymbol{\theta}_1^{\mathrm{UB}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \boldsymbol{\theta}_2^{\mathrm{UB}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}_1^{\mathrm{LB}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}_2^{\mathrm{LB}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}^{\mathrm{L}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}_1^{\mathrm{PQ}} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}_2^{\mathrm{PQ}} \end{bmatrix}_{21\times 7}, \tag{8.5c}$$

$$\boldsymbol{\theta}_1^{\mathrm{UB}} = \begin{bmatrix} -6.989 \\ +1 \end{bmatrix}_{2\times 1}, \tag{8.5d}$$

$$\boldsymbol{\theta}_2^{\mathrm{UB}} = \begin{bmatrix} -4.999 \\ +1 \end{bmatrix}_{2\times 1}, \tag{8.5e}$$

$$\boldsymbol{\theta}_1^{\mathrm{LB}} = \begin{bmatrix} +0.999 \\ +1 \end{bmatrix}_{2\times 1}, \tag{8.5f}$$

$$\boldsymbol{\theta}_2^{\mathrm{LB}} = \begin{bmatrix} +2.257\times 10^{-8} \\ +1 \end{bmatrix}_{2\times 1}, \tag{8.5g}$$

$$\boldsymbol{\theta}^{\mathrm{L}} = \begin{bmatrix} -4.798\times 10^{-1} \\ +4.995\times 10^{-2} \\ +4.438\times 10^{-2} \end{bmatrix}_{3\times 1}, \tag{8.5h}$$

$$\boldsymbol{\theta}_1^{\mathrm{PQ}} = \begin{bmatrix} -0.807 \\ +4.977 \\ +2.181 \\ -1.253 \\ -1.193 \end{bmatrix}_{5\times 1}, \tag{8.5i}$$

$$\boldsymbol{\theta}_2^{\mathrm{PQ}} = \begin{bmatrix} -1.619\times 10^{-1} \\ +6.425\times 10^{-2} \\ +5.389\times 10^{-4} \\ -6.418\times 10^{-3} \\ +1.331\times 10^{-5} \end{bmatrix}_{5\times 1}, \tag{8.5j}$$

where Equation (8.5a) shows implicit constraints in matrix form, as introduced in Chapter 7, Equation (8.5b) exhibits the augmented design matrix formed by the design sub-matrices belonging to the form-specific constitutive bound constraints (as in Equation (7.5)), the form-specific constitutive linear constraint (as in Equation (7.1)), the form-free pure quadratic constraints (as in Equation (7.2)). Equation (8.5c) displays the augmented parameters matrix composed by the parameters of the individual constitutive constraints.

Figure 8.3 presents the aggregated constraint, $\mathbb{C}(\mathbf{P})$, by the light green and the identified individual constitutive constraints by the claret red, purple, cyan, yellow,

(constitutive bound constraints), orange (constitutive linear constraint) lines as well as by the dark blue and dark green (constitutive pure quadratic constraints) over the region formed by the boundary sets of feasible (blue) and infeasible (red) points.

By just comparing the identified constitutive constraints given by Equation (8.5) with the actual inequality constraints given in Equation (8.4), it can be difficult to realize the success of our algorithm since, except for the bound constraints, the values of parameters of identified constitutive constraints are different than those of the unknown inequality constraints due to the non-uniqueness of the coefficients of inequality constraints (see Chapter 5). However, it is possible to observe from Figure 8.3 that the aggregated constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible boundary points. The zero values of $J_1$and J_2 in Table 8.3 imply this as well.

As can be remembered from the last example of Chapter 6, our algorithm has detected the same feasible region, and thus identified the unknown constraints, with the form-specific constitutive constraints introduced in Chapter 5. In this example, our algorithm detects this feasible region by the two constitutive pure quadratic constraints instead of the constitutive circular and parabolic ones.

In other words, two of the form-specific constitutive constraints used in Chapter 6 are replaced with the two form-free constitutive constraints in this example, and our algorithm can approximate the unknown circular and parabolic inequality constraints with the form-free pure quadratic constraints.

This situation also implies that our algorithm can succeed at detecting this relatively complex and non-convex feasible region with the utilization of the form-free quadratic constraint to identify the unknown circular and parabolic constraints.

However, the substitution of the constitutive pure quadratic constraint means the addition of one more parameter, i.e., the cardinality of parameters of the complete set of all constitutive constraints becomes $|\boldsymbol{\theta}| = 19$ when the constitutive quadratic constraints are used.

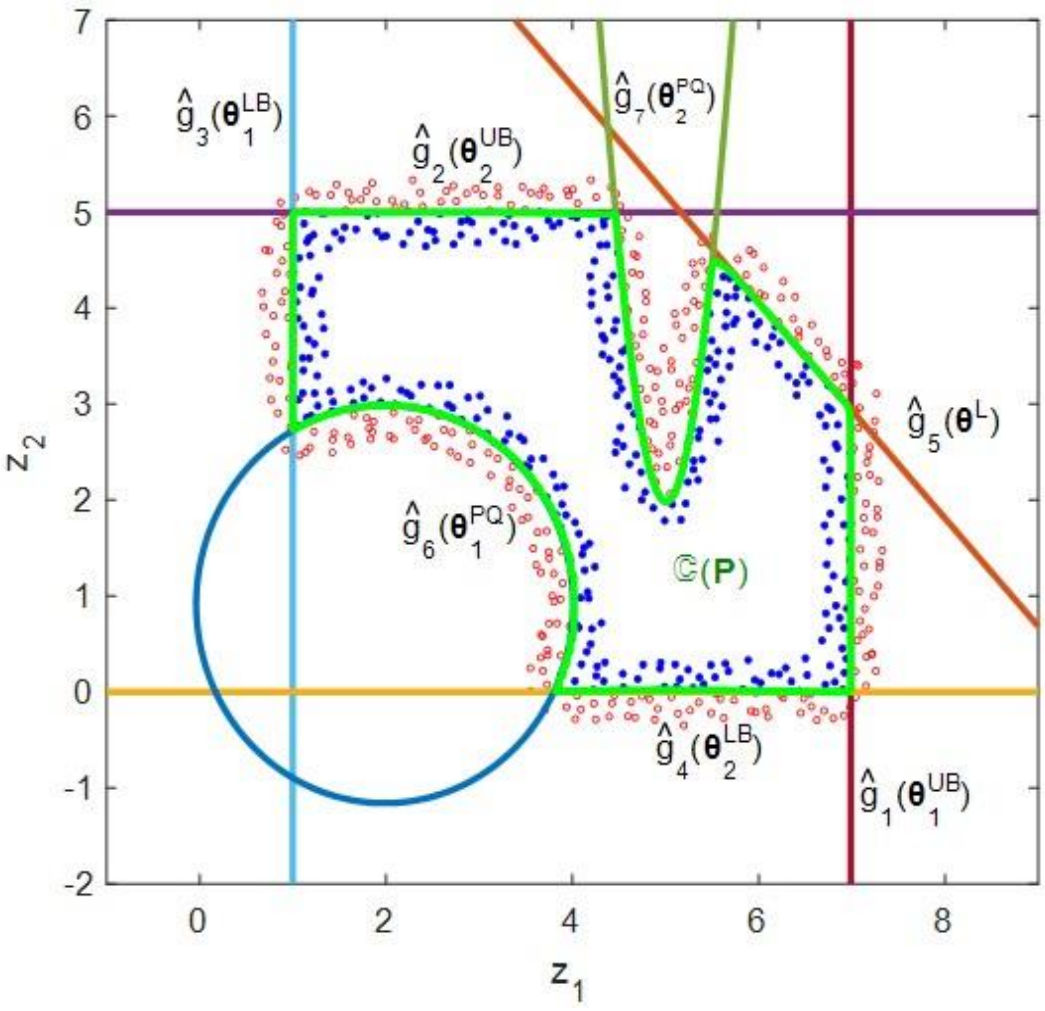

Figure 8.3. Identified optimal constitutive constraints and aggregated constraint superimposed with the boundary sets of feasible and infeasible points forming a non-convex region.

8.3. A Disjoint Infeasible Region in 2-D

In this section, we will demonstrate, for the first time, the ability of our algorithm to detect a disjoint infeasible region forming a non-convex feasible region. Six inequality constraints which will be treated as unknowns and will be detected by our algorithm are given by the following equations:

$$g_1: \quad +z_1 - 7 \leq 0, \tag{8.6a}$$

$$g_2: \quad -z_1 + 1 \leq 0, \tag{8.6b}$$

$$g_3: \quad +z_2 - 5 \leq 0, \tag{8.6c}$$

$$g_4: \quad -z_2 \leq 0, \tag{8.6d}$$

$$g_5: \quad +z_1 + z_2 - 10 \leq 0, \tag{8.6e}$$

$$g_6: \quad -(z_1 - 4)^2 - (z_2 - 2)^2 + 0.9 \leq 0. \tag{8.6f}$$

SLHS is employed to generate sample points. For this example, information on the sampling and optimization, as well as the results are all presented in Table 8.4.

Table 8.4. Sampling and optimization information and the results.

Sampling Information			
$\mathbf{z}^{LB} = [-1 \ -2]$	$\mathbf{z}^{UB} = [+9 \ +7]$		
$N = 2500$	$N^f = 700$		$N^i = 1800$
$\mathbf{Z} \in \mathbb{R}^{2500\times2}$	$\mathbf{X} \in \mathbb{R}^{700\times2}$		$\mathbf{Y} \in \mathbb{R}^{1800\times2}$
	$p = 1$		
$N^b = 583$	$N^{fb} = 292$		$N^{ib} = 291$
$\mathbf{Z} \in \mathbb{R}^{583\times2}$	$\mathbf{X}^b \in \mathbb{R}^{292\times2}$		$\mathbf{Y}^b \in \mathbb{R}^{291\times2}$
Optimization Settings			
$N^B = 4$	$N^L = 1$	$N^{PQ} = 1$	
$N^{Con} = 6$	$D = 2$ $\quad \lvert\boldsymbol{\theta}\rvert = 12$	$\mu = 10^{-6}$	$\rho = 500$
$\mathbf{z}^L = [-10 \ -10]$	$\mathbf{z}^U = [+10 \ +10]$	$NP = 40$	$Tol = 10^{-9}$
Optimization Results			
Iterations $= 3691$	Function Evaluations $= 112980$		
$J = 13.068 \times 10^{-6}$	$J_1 = 0$	$J_2 = 0$	$J_3 = 13.068$
Perimeter $= 26.751$	Area $= 24.975$		

The constitutive inequality constraints identified and the optimal values of parameters are all given as:

$$\begin{bmatrix} \hat{\mathbf{g}}_1^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}_1^{\mathrm{UB}}) \\ \cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_2^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}_2^{\mathrm{UB}}) \\ \cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_3^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}_1^{\mathrm{LB}}) \\ \cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_4^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}_2^{\mathrm{LB}}) \\ \cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_5^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}^{\mathrm{L}}) \\ \cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_6^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}^{\mathrm{PQ}}) \end{bmatrix}^{\mathrm{T}} \coloneqq \boldsymbol{\Phi}(\mathbf{Z})\mathbf{P} = \left(\begin{bmatrix} (\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_1)\boldsymbol{\theta}_1^{\mathrm{UB}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_2)\boldsymbol{\theta}_2^{\mathrm{UB}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_1)\boldsymbol{\theta}_1^{\mathrm{LB}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_2)\boldsymbol{\theta}_2^{\mathrm{LB}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})\boldsymbol{\theta}^{\mathrm{L}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})\boldsymbol{\theta}^{\mathrm{PQ}})^{\mathrm{T}} \end{bmatrix}_{6\times 583} \right)^{\mathrm{T}}, \tag{8.7a}$$

$$\boldsymbol{\Phi}(\mathbf{Z}) = \left(\begin{bmatrix} \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_1)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_2)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_1)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_2)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{L}}(\mathbf{Z})\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})\right)^{\mathrm{T}} \end{bmatrix}_{16\times 583} \right)^{\mathrm{T}}, \tag{8.7b}$$

$$\mathbf{P} = \begin{bmatrix} \boldsymbol{\theta}_1^{\mathrm{UB}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \boldsymbol{\theta}_2^{\mathrm{UB}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}_1^{\mathrm{LB}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}_2^{\mathrm{LB}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}^{\mathrm{L}} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}^{\mathrm{PQ}} \end{bmatrix}_{16\times 6}, \tag{8.7c}$$

$$\boldsymbol{\theta}_1^{\mathrm{UB}} = \begin{bmatrix} -6.999 \\ +1 \end{bmatrix}_{2\times 1}, \tag{8.7d}$$

$$\boldsymbol{\theta}_2^{\mathrm{UB}} = \begin{bmatrix} -4.993 \\ +1 \end{bmatrix}_{2\times 1}, \tag{8.7e}$$

$$\boldsymbol{\theta}_1^{\mathrm{LB}} = \begin{bmatrix} +0.994 \\ +1 \end{bmatrix}_{2\times 1}, \tag{8.7f}$$

$$\boldsymbol{\theta}_2^{\mathrm{LB}} = \begin{bmatrix} -3.194\times 10^{-8} \\ +1 \end{bmatrix}_{2\times 1}, \tag{8.7g}$$

$$\boldsymbol{\theta}^{\mathrm{L}} = \begin{bmatrix} -3.582\times 10^{-2} \\ +3.492\times 10^{-3} \\ +3.731\times 10^{-3} \end{bmatrix}_{3\times 1}, \tag{8.7h}$$

$$\boldsymbol{\theta}^{\mathrm{PQ}} = \begin{bmatrix} -2.175 \times 10^{-2} \\ +9.040 \times 10^{-3} \\ +4.622 \times 10^{-3} \\ -1.124 \times 10^{-3} \\ -1.157 \times 10^{-3} \end{bmatrix}_{5\times 1}, \tag{8.7i}$$

where Equation (8.7a) shows implicit constraints in matrix form, as introduced in Chapter 7, Equation (8.7b) exhibits the augmented design matrix formed by the design sub-matrices belonging to the form-specific constitutive bound constraints (as in Equation (7.5)), the form-specific constitutive linear constraint (as in Equation (7.1)), the form-free pure quadratic constraint (as in Equation (7.2)). Equation (8.7c) displays the parameters matrix created by augmenting the parameters of these types of constitutive constraints.

Figure 8.4 presents the aggregated constraint, $\mathbb{C}(\mathbf{P})$, by the light green and the identified individual constitutive constraints by the claret red, purple, cyan, yellow, (constitutive bound constraints), orange (constitutive linear constraint) lines as well as by the light green circle (constitutive pure quadratic constraint) over the region formed by the boundary sets of feasible (blue) and infeasible (red) points.

By just comparing the identified constitutive constraints given by Equation (8.7) with the actual inequality constraints given in Equation (8.6), it can be difficult to realize the success of our algorithm since, except for the bound constraints, the values of parameters of identified constitutive constraints are different than those of the unknown inequality constraints due to the non-uniqueness of the coefficients of inequality constraints (see Chapter 5).

However, it is possible to observe from Figure 8.4 that the aggregated constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible boundary points. The zero values of J_1 and J_2 in Table 8.4 imply this as well.

Thus, this example proves that our algorithm can identify disjoint regions as well. Had we used two or more circular (or any other shape) infeasible regions (overlapping or nonoverlapping with each other) within the feasible region, our algorithm would again have correctly identified the disjoint circular regions disconnected from the infeasible points outside the polygonal region.

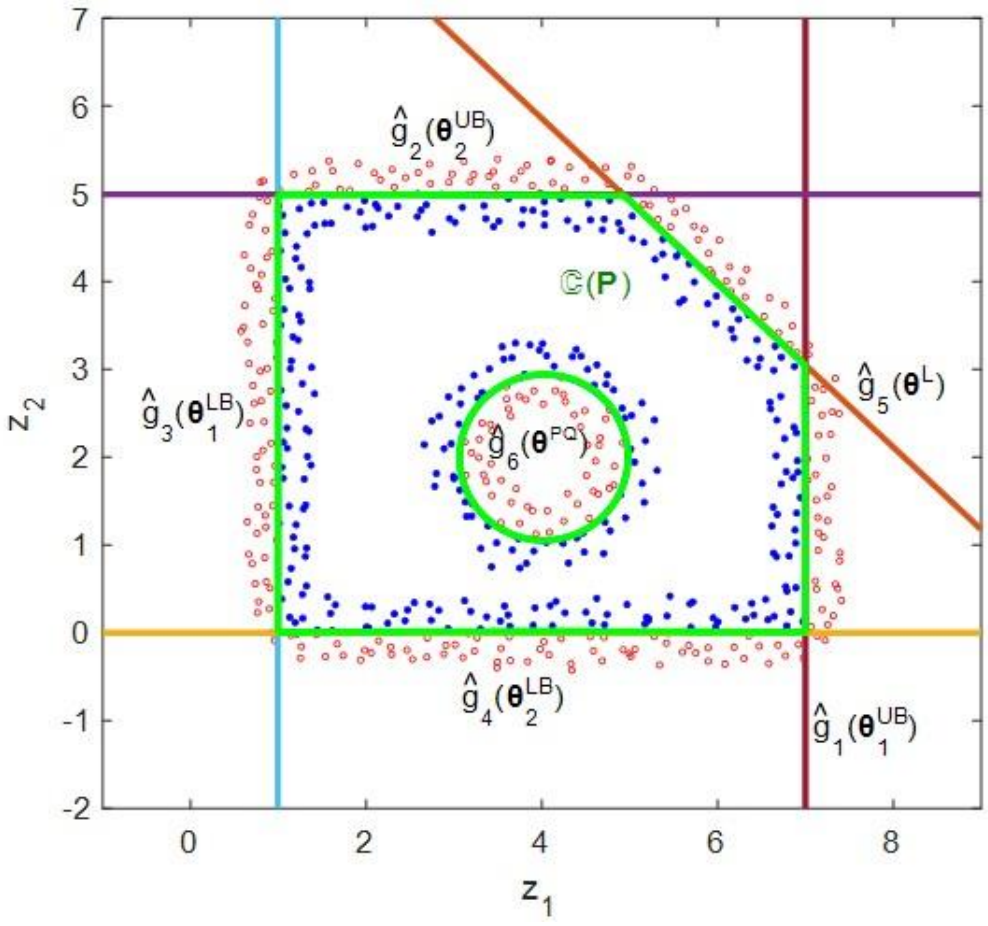

Figure 8.4. Identified optimal constitutive constraints and aggregated constraint superimposed with the boundary sets of feasible and infeasible points forming a non-convex feasible region and disjoint infeasible region.

8.4. A Fragmented Disjoint Feasible Region in 2-D

In this section, we will demonstrate, for the first time, the ability of our algorithm to detect fragmented (multiple) disjoint feasible regions forming non-convex feasible and non-convex infeasible regions. Two inequality constraints which will be treated as unknowns and will be detected by our algorithm are given by the following equations:

$$g_1: \quad -0.05(z_1-4)^2-0.1(z_2-2.5)^2+0.2(z_1-4)(z_2-2.5)+1\leq 0, \qquad (8.8a)$$

$$g_2: \quad -0.05(z_1-4)^2-0.1(z_2-2.5)^2-0.2(z_1-4)(z_2-2.5)+1\leq 0. \qquad (8.8b)$$

SLHS is employed to generate sample points. For this example, information on the sampling and optimization, as well as the results are all presented in Table 8.5.

Table 8.5. Sampling and optimization information and the results.

Sampling Information				
$\mathbf{z}^{\mathrm{LB}} = [-4 \; -6]$	$\mathbf{z}^{\mathrm{UB}} = [+18 \; +20]$			
$\mathrm{N} = 5000$	$\mathrm{N}^{\mathrm{f}} = 1876$	$\mathrm{N}^{\mathrm{i}} = 3165$		
$\mathbf{Z} \in \mathbb{R}^{5000\times 2}$	$\mathbf{X} \in \mathbb{R}^{1876\times 2}$	$\mathbf{Y} \in \mathbb{R}^{3165\times 2}$		
	$\mathrm{p} = 1$			
$\mathrm{N}^{\mathrm{b}} = 1398$	$\mathrm{N}^{\mathrm{fb}} = 653$	$\mathrm{N}^{\mathrm{ib}} = 745$		
$\mathbf{Z} \in \mathbb{R}^{1398\times 2}$	$\mathbf{X}^{\mathrm{b}} \in \mathbb{R}^{653\times 2}$	$\mathbf{Y}^{\mathrm{b}} \in \mathbb{R}^{745\times 2}$		
Optimization Settings				
$\mathrm{N}^{\mathrm{Con}} = \mathrm{N}^{\mathrm{Q}} = 2$	$\mathrm{D} = 2$	$\lvert\boldsymbol{\theta}\rvert = 12$	$\mu = 10^{-6}$	$\rho = 500$
$\mathbf{z}^{\mathrm{L}} = [-10 \; -10]$	$\mathbf{z}^{\mathrm{U}} = [+10 \; +10]$	$\mathrm{NP} = 40$	$\mathrm{Tol} = 10^{-9}$	
Optimization Results				
$\mathrm{Iterations} = 18755$	$\mathrm{Function\ Evaluations} = 709363$			
$\mathrm{J} = 6.952 \times 10^{-8}$	$\mathrm{J}_1 = 0$	$\mathrm{J}_2 = 0$	$\mathrm{J}_3 = 0.069$	
$\mathrm{Perimeter} = 108.587$	$\mathrm{Area} = 198.398$			

The form-free constitutive quadratic inequality constraints identified and the optimal values of parameters are all given by:

$$\begin{bmatrix} \hat{\mathbf{g}}_1^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_1^{\mathrm{Q}}) \\ \cdots\cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_2^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_2^{\mathrm{Q}}) \end{bmatrix}^{\mathrm{T}} \coloneqq \boldsymbol{\Phi}(\mathbf{Z})\mathbf{P} = \left(\begin{bmatrix} \left(\boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{Z})\boldsymbol{\theta}_1^{\mathrm{Q}}\right)^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots\cdots \\ \left(\boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{Z})\boldsymbol{\theta}_2^{\mathrm{Q}}\right)^{\mathrm{T}} \end{bmatrix}_{2\times 1398} \right)^{\mathrm{T}}, \tag{8.9a}$$

$$\boldsymbol{\Phi}(\mathbf{Z}) = [\boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{Z}) \quad \boldsymbol{\Phi}^{\mathrm{Q}}(\mathbf{Z})]_{1398\times 12}, \tag{8.9b}$$

$$\mathbf{P} = \begin{bmatrix} \boldsymbol{\theta}_1^{\mathrm{Q}} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \boldsymbol{\theta}_2^{\mathrm{Q}} \end{bmatrix}_{12\times 2}, \tag{8.9c}$$

$$\boldsymbol{\theta}_1^{\mathrm{Q}} = \begin{bmatrix} +2.058 \times 10^{-2} \\ -1.364 \times 10^{-3} \\ -3.886 \times 10^{-3} \\ +2.606 \times 10^{-3} \\ -6.492 \times 10^{-4} \\ -1.302 \times 10^{-3} \end{bmatrix}_{6\times 1}, \tag{8.9d}$$

$$\boldsymbol{\theta}_1^{\mathrm{Q}} = \begin{bmatrix} -1.907 \times 10^{-2} \\ +7.088 \times 10^{-3} \\ +1.021 \times 10^{-2} \\ -1.572 \times 10^{-3} \\ -3.941 \times 10^{-4} \\ -7.851 \times 10^{-4} \end{bmatrix}_{6\times 1}, \tag{8.9e}$$

where Equation (8.9a) shows implicit constraints in matrix form, as introduced in Chapter 7, Equation (8.9b) exhibits the augmented design matrix formed by the design sub-matrices belonging to the form-specific constitutive quadratic constraint (as in Equation (7.4)). Equation (8.9c) displays the parameters matrix created by augmenting the parameters of this type of constitutive constraints.

Figure 8.5 presents the aggregated constraint, $\mathbb{C}(\mathbf{P})$, by the light green, and the identified individual constitutive constraints by the claret red, cyan over the region formed by the boundary sets of feasible (blue) and infeasible (red) points. By just comparing the identified constitutive constraints given by Equation (8.9) with the actual inequality constraints given in Equation (8.8), it can be difficult to realize the success of our algorithm since the values of parameters of identified constitutive constraints are different than those of the unknown inequality constraints due to the non-uniqueness of the coefficients of inequality constraints (see Chapter 5). However, it is possible to observe from Figure 8.5 that the aggregated constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible boundary points. The zero values of $\mathrm{J}_1$and J_2 in Table 8.5 imply this as well.

Thus, this example proves that our algorithm can also identify multiple disjoint feasible and infeasible regions.

It should be noticed from the figure that we are using two full-quadratic constitutive constraints to aggregate four different feasible regions (and one infeasible region). In other words, to aggregate four disjoint feasible regions we do not need to use four constitutive constraints. In the figure it can be seen that each branch curve (there are two branches with identical $\mathbb{C}(\mathbf{P}) = 0$ contour values) of each one of the constitutive constraints serves in aggregation of two disjoint feasible regions coincidently.

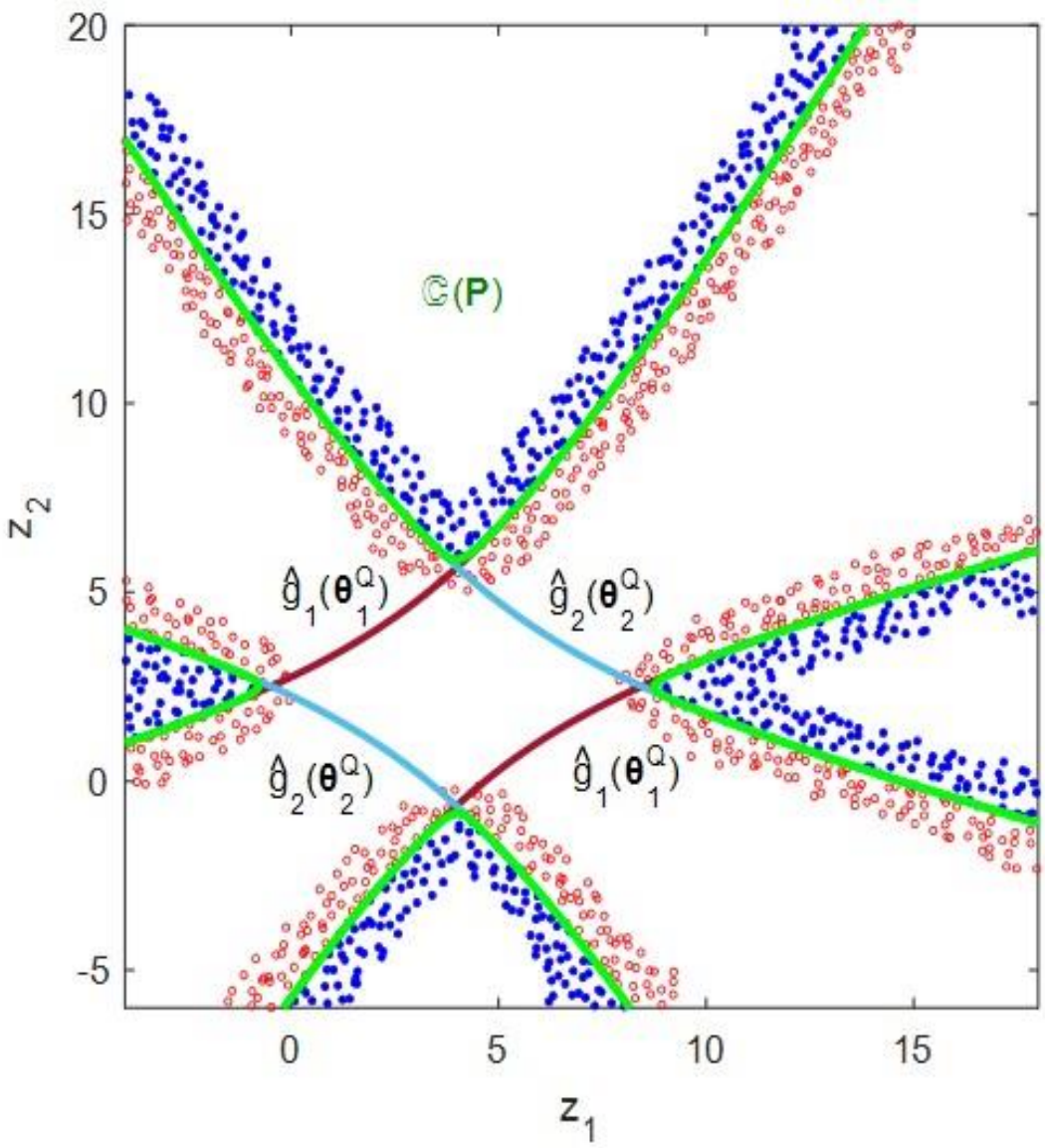

Figure 8.5. Identified optimal constitutive constraints and aggregated constraint superimposed with the boundary sets of feasible and infeasible points forming fragmented disjoint feasible and infeasible regions.

8.5. A 3-D Non-Convex Region

In this section, we will demonstrate the ability of our algorithm to detect a non-convex, three-dimensional feasible region formed by bound constraints and a spherical inequality constraint via the design matrix approach. Seven inequality constraints which will be treated as unknowns and will be detected by our algorithm are given by the following equation:

$$g_1: \quad +z_1 - 2 \leq 0, \tag{8.10a}$$

$$g_2: \quad -z_1 \leq 0, \tag{8.10b}$$

$$g_3: \quad +z_2 - 2 \leq 0, \tag{8.10c}$$

$$g_4: \quad -z_2 \leq 0, \tag{8.10d}$$

$$g_5: \quad +z_3 - 2 \leq 0, \tag{8.10e}$$

$$g_6: \quad -z_3 \leq 0, \tag{8.10f}$$

$$g_7: \quad -(z_1)^2 - (z_2 - 1)^2 - (z_3 - 1.5)^2 + 2.5 \leq 0. \tag{8.10g}$$

SLHS is employed to generate sample points. For this example, information on the sampling and optimization, as well as the results are all presented in Table 8.6

Table 8.6. Sampling and optimization information and the results.

Sampling Information			
$\mathbf{z}^{LB} = [-1 \ -1 -1]$	$\mathbf{z}^{UB} = [+3 \ +3 +3]$		
$N = 6000$	$N^f = 342$		$N^i = 6517$
$\mathbf{Z} \in \mathbb{R}^{6000\times3}$	$\mathbf{X} \in \mathbb{R}^{342\times3}$		$\mathbf{Y} \in \mathbb{R}^{6517\times3}$
	$p = 1$		
$N^b = 653$	$N^{fb} = 288$		$N^{ib} = 365$
$\mathbf{Z} \in \mathbb{R}^{653\times3}$	$\mathbf{X}^b \in \mathbb{R}^{288\times3}$		$\mathbf{Y}^b \in \mathbb{R}^{365\times3}$
Optimization Settings			
$N^B = 6$	$N^{PQ} = 1$		
$N^{Con} = 7$	$D = 3$	$\lvert\boldsymbol{\theta}\rvert = 13$	$\mu = 10^{-6}$ $\rho = 500$
$\mathbf{z}^L = [-10 \ -10 -10]$	$\mathbf{z}^U = [+10 \ +10 +10]$	$NP = 40$	$Tol = 10^{-9}$
Optimization Results			
Iterations $= 9911$	Function Evaluations $= 324869$		
$J = 6.093 \times 10^{-6}$	$J_1 = 0$	$J_2 = 0$	$J_3 = 6.093$

The constitutive inequality constraints identified and the optimal values of parameters are all given by:

$$\boldsymbol{\Phi}(\mathbf{Z})\mathbf{P} = \begin{bmatrix} \hat{\mathbf{g}}_1^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_1^{\mathrm{UB}}) \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_2^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_2^{\mathrm{UB}}) \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_3^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_3^{\mathrm{UB}}) \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_4^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_1^{\mathrm{LB}}) \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_5^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_2^{\mathrm{LB}}) \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_6^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}_3^{\mathrm{LB}}) \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_7^{\mathrm{T}}(\mathbf{Z}, \boldsymbol{\theta}^{\mathrm{PQ}}) \end{bmatrix}^{\mathrm{T}}$$

$$= \left(\begin{bmatrix} \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_1)\boldsymbol{\theta}_1^{\mathrm{UB}}\right)^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_2)\boldsymbol{\theta}_2^{\mathrm{UB}}\right)^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_3)\boldsymbol{\theta}_3^{\mathrm{UB}}\right)^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_1)\boldsymbol{\theta}_1^{\mathrm{LB}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_2)\boldsymbol{\theta}_2^{\mathrm{LB}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_3)\boldsymbol{\theta}_3^{\mathrm{LB}})^{\mathrm{T}} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ (\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})\boldsymbol{\theta}^{\mathrm{PQ}})^{\mathrm{T}} \end{bmatrix}_{7\times 653} \right)^{\mathrm{T}}, \tag{8.11a}$$

$$\boldsymbol{\Phi}(\mathbf{Z}) = \left(\begin{bmatrix} \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_1)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_2)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{UB}}(\mathbf{z}_3)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_1)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_2)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{LB}}(\mathbf{z}_3)\right)^{\mathrm{T}} \\ \left(\boldsymbol{\Phi}^{\mathrm{PQ}}(\mathbf{Z})\right)^{\mathrm{T}} \end{bmatrix}_{19\times 635} \right)^{\mathrm{T}}, \tag{8.11b}$$

$$\mathbf{P} = \begin{bmatrix} \boldsymbol{\theta}_1^{\mathrm{UB}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \boldsymbol{\theta}_2^{\mathrm{UB}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}_3^{\mathrm{UB}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}_1^{\mathrm{LB}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}_2^{\mathrm{LB}} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}_3^{\mathrm{LB}} & \vdots & \mathbf{0} \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{\theta}^{\mathrm{PQ}} \end{bmatrix}_{19\times 7}, \quad (8.11c)$$

$$\boldsymbol{\theta}_1^{\mathrm{UB}} = \begin{bmatrix} -1.997 \\ 1 \end{bmatrix}_{2\times 1}, \quad (8.11d)$$

$$\boldsymbol{\theta}_2^{\mathrm{UB}} = \begin{bmatrix} -1.999 \\ 1 \end{bmatrix}_{2\times 1}, \quad (8.11e)$$

$$\boldsymbol{\theta}_3^{\mathrm{UB}} = \begin{bmatrix} -1.992 \\ 1 \end{bmatrix}_{2\times 1}, \quad (8.11f)$$

$$\boldsymbol{\theta}_1^{\mathrm{LB}} = \begin{bmatrix} 1.095 \times 10^{-9} \\ 1 \end{bmatrix}_{2\times 1}, \quad (8.11g)$$

$$\boldsymbol{\theta}_2^{\mathrm{LB}} = \begin{bmatrix} 7.115 \times 10^{-9} \\ 1 \end{bmatrix}_{2\times 1}, \quad (8.11h)$$

$$\boldsymbol{\theta}_3^{\mathrm{LB}} = \begin{bmatrix} 2.617 \times 10^{-9} \\ 1 \end{bmatrix}_{2\times 1}, \quad (8.11i)$$

$$\boldsymbol{\theta}^{\mathrm{PQ}} = \begin{bmatrix} -8.000 \times 10^{-3} \\ -7.585 \times 10^{-4} \\ +2.236 \times 10^{-2} \\ +3.333 \times 10^{-2} \\ -1.082 \times 10^{-2} \\ -1.114 \times 10^{-2} \\ -1.104 \times 10^{-2} \end{bmatrix}_{7\times 1}, \quad (8.11j)$$

where Equation (8.11a) shows implicit constraints in matrix form, as introduced in Chapter 7, Equation (8.11b) exhibits the augmented design matrix formed by the design sub-matrices belonging to the form-specific constitutive bound constraints (as in Equation (7.5)), the form-free pure quadratic constraint (as in Equation (7.2)). Equation (8.11c) displays the parameters matrix created by augmenting the parameters of these types of constitutive constraints.

Figure 8.6a presents the aggregated three-dimensional constraint, $\mathbb{C}(\mathbf{P})$, by the black mesh, (a cage that encloses all the blue feasible points), and the identified individual constitutive constraints by the red, blue, yellow, green, cyan, magenta planes (constitutive bound constraints), the white sphere (constitutive pure quadratic constraint) together with

the three-dimensional region formed by the boundary sets of feasible (blue) and infeasible (red) points. Figure 8.6b-d show the projections of Figure 8.6a onto all pairs of its binary planes. By just comparing the identified constitutive constraints given by Equation (8.11) with the actual inequality constraints given in Equation (8.10), it can be difficult to realize the success of our algorithm since, except for the bound constraints, the values of parameters of identified constitutive constraints are different than those of the unknown inequality constraints due to the non-uniqueness of the coefficients of inequality constraints (see Chapter 5). However, it is possible to observe from Figure 8.6 that the aggregated constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible boundary points. The zero values of J_1 and J_2 in Table 8.6 imply this as well.

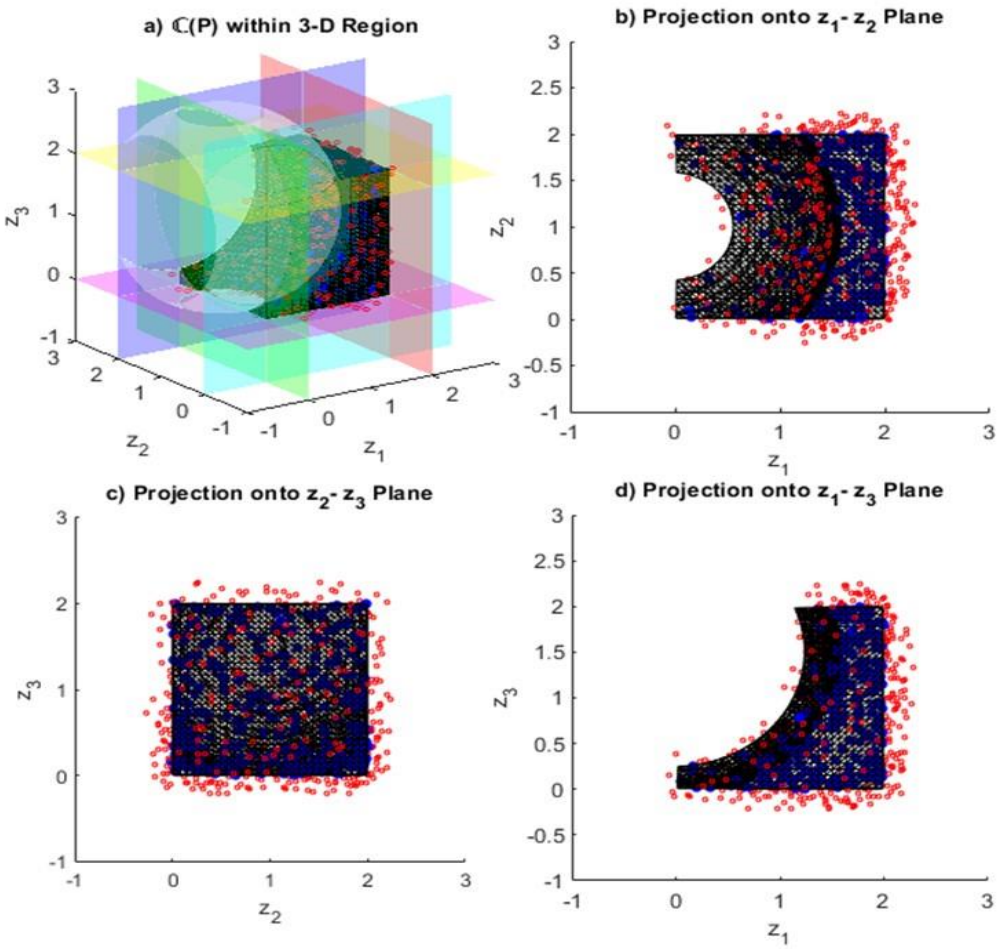

Figure 8.6. Identified optimal constitutive constraints and aggregated constraint superimposed with the boundary sets of feasible and infeasible points forming a three-dimensional non-convex region.

8.6. Single High-Order Polynomial Constitutive for a Non-Convex Region in 2-D

In this section, we will demonstrate our algorithm's flexibility in utilizing high-order polynomials as constitutive constraints to detect a non-convex region. Instead of the form-specific and form free constitutive constraints with at most full-quadratic (since the options of MATLAB's "x2fx" function allow us to build only linear, interaction, pure quadratic, and quadratic models), our algorithm can utilize a single but higher order model to build a form-free constitutive constraint. We still use MATLAB's "x2fx" function, but with its more flexible "model" option. It should be noted that this option allows only definition of powers for the factors (the columns of our design matrix that are also called "effects"). By utilizing MATLAB's x2fx function with "model" option together with full-factorial design we can build models with orders greater than two. By this way, we are not limited by full-quadratic, but can generate much higher-order models (e.g., 3rd, 4th, 5th order models). Such models include $(\mathrm{m}+1)^{\mathrm{D}}$ columns where m denotes the order of model (e.g., $\mathrm{m}=1$ is linear model, $\mathrm{m}=2$ is quadratic model, etc.), and D implies the dimension of the coordinate system. The total number of optimization parameters in constraint identification is $(\mathrm{m}+1)^{\mathrm{D}}$, i.e., $\boldsymbol{\theta} \in \mathbb{R}^{(\mathrm{m}+1)^{\mathrm{D}}}$ For instance, full-factorial design for the 3rd order, two-dimensional model, i.e., $\mathrm{m}=3$ and $\mathrm{D}=2$, will have the 4^2 columns and these columns corresponds to the 16 factors given in Table 8.7:

It should be also highlighted that there is no need for the KS aggregation function when there is a single constitutive constraint model such as only one high-order polynomial.

The six inequality constraints which will be treated as unknowns and will be detected by our algorithm are given by the following equations:

$$\mathrm{g}_1: \quad +\mathrm{z}_1 - 7 \leq 0, \tag{8.12a}$$

$$\mathrm{g}_2: \quad -\mathrm{z}_1 + 1 \leq 0, \tag{8.12b}$$

$$\mathrm{g}_3: \quad +\mathrm{z}_2 - 5 \leq 0, \tag{8.12c}$$

$$\mathrm{g}_4: \quad -\mathrm{z}_2 \leq 0, \tag{8.12d}$$

$$\mathrm{g}_5: \quad +\mathrm{z}_1 + \mathrm{z}_2 - 10 \leq 0, \tag{8.12e}$$

$$g_6: \quad -(z_1 - 2)^2 - (z_2 - 1)^2 + 4 \leq 0. \tag{8.12f}$$

In this example, we approximate the above six inequality constraints forming the feasible region via a single 4th-order polynomial constitutive constraint. Also, in this example, the boundary-zone formation procedure was not used since some zero-valued contours may emerge in some part of the non-sampled region, i.e., some part of the region beyond the boundary zone which does not include any feasible or infeasible points, even though our algorithm can find a single 4th-order polynomial constitutive constraint satisfying the constraints. Thus, MS is employed to generate the full set of sample points (as outlined in Chapter 3).

Table 8.7. The factors of the design matrix with full-factorial design (m=3 and D=2).

Factors		**Effect**
$(z_1$	$z_2)$	
0	0	Constant
1	0	z_1
2	0	z_1^2
3	0	z_1^3
0	1	z_2
1	1	$z_1 z_2$
2	1	$z_1^2 z_2$
3	1	$z_1^3 z_2$
0	2	z_2^2
1	2	$z_1 z_2^2$
2	2	$z_1^2 z_2^2$
3	2	$z_1^3 z_2^2$
0	3	z_2^3
1	3	$z_1^1 z_2^3$
2	3	$z_1^2 z_2^3$
3	3	$z_1^3 z_2^3$

Information on the sampling and optimization, as well as the results are all presented in Table 8.8.

The form-free 4th-order polynomial constitutive constraint identified is given explicitly, together with the optimal values of parameters, by the following equation:

$$\begin{aligned}\hat{g}(\boldsymbol{\theta}) = \; & 2.834 - 1.525z_1 + 0.576z_1^2 - 0.147z_1^3 + 0.012z_1^4 \\ & - 4.018z_2 + 0.793z_1z_2 + 2.348z_1^2z_2 - 0.900z_1^3z_2 \\ & + 0.077z_1^4z_2 + 1.154z_2^2 + 1.010z_1z_2^2 + 0.042z_1^2z_2^2 \\ & - 0.098z_1^3z_2^2 + 0.013z_1^4z_2^2 + 0.263z_2^3 - 1.168z_1z_2^3 \\ & - 0.134z_1^2z_2^3 + 0.133z_1^3z_2^3 - 0.015z_1^4z_2^3 - 0.018z_2^4 \\ & + 0.088z_1z_2^4 + 0.069z_1^2z_2^4 - 0.031z_1^3z_2^4 + 0.003z_1^4z_2^4.\end{aligned} \tag{8.13}$$

Table 8.8. Sampling and optimization information and the results.

Sampling Information				
$\mathbf{z}^{\mathrm{LB}} = [-1 \;\; -2]$	$\mathbf{z}^{\mathrm{UB}} = [+9 \;\; +7]$			
$\mathrm{N} = 2000$	$\mathrm{N}^{\mathrm{f}} = 450$			$\mathrm{N}^{\mathrm{i}} = 1575$
$\mathbf{Z} \in \mathbb{R}^{2000\times 2}$	$\mathbf{X} \in \mathbb{R}^{450\times 2}$			$\mathbf{Y} \in \mathbb{R}^{1575\times 2}$
Optimization Settings				
$\mathrm{N}^{\mathrm{Con}} = 1$	$\mathrm{m} = 4$	$\mathrm{D} = 2$	$\lvert\boldsymbol{\theta}\rvert = 25$	$\mu = 10^{-6}$
$\mathbf{z}^{\mathrm{L}} = [-5 \;\; -5]$	$\mathbf{z}^{\mathrm{U}} = [+5 \;\; +5]$		$\mathrm{NP} = 150$	$\mathrm{Tol} = 10^{-9}$
Optimization Results				
$\mathrm{Iterations} = 45305$	$\mathrm{Function\ Evaluations} = 4095108$			
$\mathrm{J} = 1.747 \times 10^{-5}$	$\mathrm{J}_1 = 0$	$\mathrm{J}_2 = 0$	$\mathrm{J}_3 = 17.478$	
$\mathrm{Perimeter} = 20.535$	$\mathrm{Area} = 21.245$			

Figure 8.7 presents this constitutive constraint, $\hat{g}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour) by the light green over the region formed by the complete sets of feasible (blue) and infeasible (red) points. There is no occasion to judge the success of our algorithm by just comparing the identified constitutive constraint given by Equation (8.13) with the unknown or actual constraints given in Equation (8.12) since we are approximating the feasible region formed by the six constraints via a 4th-order polynomial constitutive

constraint. However, it is possible to observe from Figure 8.7 that the identified constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. The zero values of J_1 and J_2 in Table 8.8 imply this as well.

By increasing the number of sample points, our algorithm could have achieved less curly constraint curve. However, constraint identification becomes increasingly difficult in that case. Thus, we decreased the number of sample points and used more sparse sample points to ease the optimization task. This point is important and opens an idea for further research, i.e., constraint identification under uncertainty (uncertainty in identified constraint(s) due to excess space among the sparse sample points, as observed in Figure 8.7 below). However, this topic is outside the scope of this thesis work.

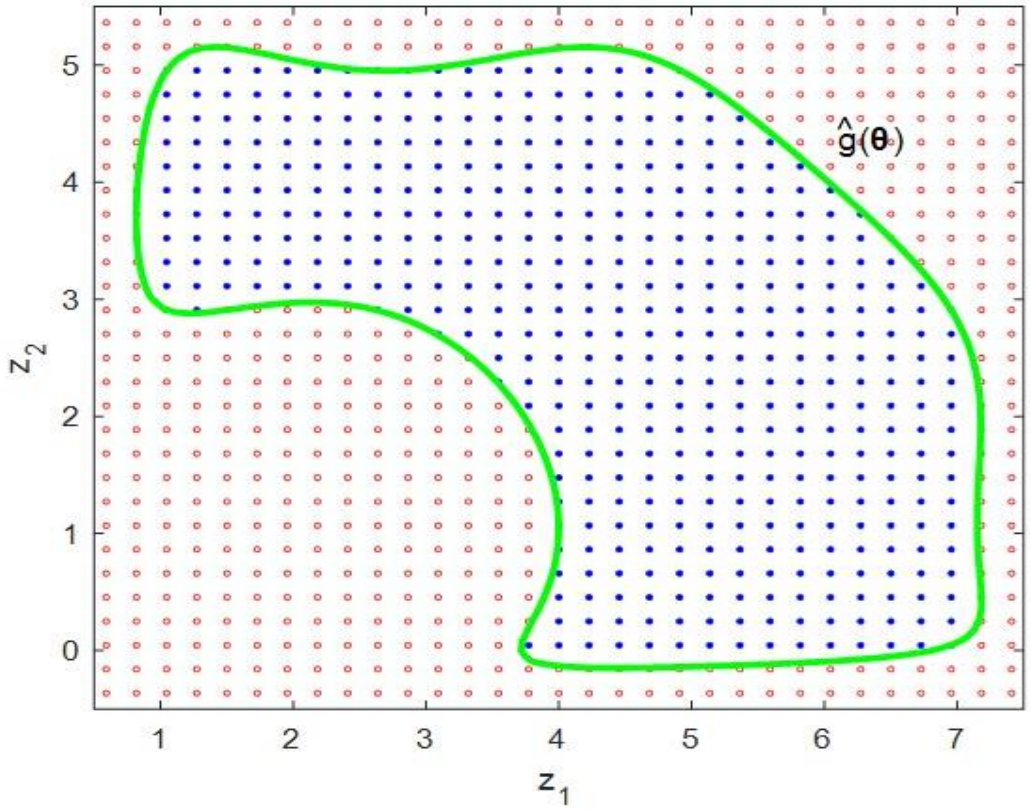

Figure 8.7. Identified optimal single 4th-order polynomial constitutive constraint superimposed with the complete sets of feasible and infeasible points forming a non-convex region.

8.7. The Use of Single Yet High-Order Constitutive Polynomials for Constraint Identification

In this section, our algorithm will approximate the feasible regions studied in Section 8.3 and Section 8.2 via single high-order polynomial constitutive constraints.

For the first example of this section, the six inequality constraints which will be treated as unknowns and will be detected by our algorithm had already been presented in Section 8.3 with Equation (8.6).

With the first example of this section, we approximate the six inequality constraints forming the disjoint infeasible region via a single 5^{th}-order polynomial constitutive constraint. Also, in this part, the boundary-zone formation procedure was not used as reasoned in the previous section. Thus, MS is employed to generate the full set of sample points as outlined in Chapter 3. Information on the sampling and optimization, as well as the results are all presented in Table 8.9.

Table 8.9. Sampling and optimization information and the results.

Sampling Information				
$\mathbf{z}^{\mathrm{LB}} = [-1 \ -2]$	$\mathbf{z}^{\mathrm{UB}} = [+9 \ +7]$			
$\mathrm{N} = 3000$	$\mathrm{N^f} = 857$			$\mathrm{N^i} = 2168$
$\mathbf{Z} \in \mathbb{R}^{3000\times 2}$	$\mathbf{X} \in \mathbb{R}^{857\times 2}$			$\mathbf{Y} \in \mathbb{R}^{2168\times 2}$
Optimization Settings				
$\mathrm{N^{Con}} = 1$	$\mathrm{m} = 5$	$\mathrm{D} = 2$	$\lvert\boldsymbol{\theta}\rvert = 36$	$\mu = 10^{-6}$
$\mathbf{z}^{\mathrm{L}} = [-5 \ -5]$	$\mathbf{z}^{\mathrm{U}} = [+5 \ +5]$		$\mathrm{NP} = 150$	$\mathrm{Tol} = 10^{-9}$
Optimization Results				
$\mathrm{Iterations} = 16266$	$\mathrm{Function\ Evaluations} = 2287780$			
$\mathrm{J} = 2.861 \times 10^{-5}$	$\mathrm{J}_1 = 0$	$\mathrm{J}_2 = 0$	$\mathrm{J}_3 = 28.616$	
$\mathrm{Perimeter} = 27.054$	$\mathrm{Area} = 26.69$			

The form-free 5th-order polynomial constitutive constraint identified is given explicitly, together with the optimal values of parameters, by the following equation:

$$\begin{aligned}\hat{g}(\boldsymbol{\theta}) = \; & 2.321 - 1.588z_1 - 1.610z_1^2 + 1.177z_1^3 - 0.243z_1^4 \\ & + 0.015z_1^5 + 2.057z_2 - 2.075z_1z_2 - 4.076z_1^2z_2 \\ & + 1.712z_1^3z_2 - 0.250z_1^4z_2 + 0.012z_1^5z_2 + 3.138z_2^2 \\ & - 2.191z_1z_2^2 + 1.210z_1^2z_2^2 + 1.307z_1^3z_2^2 - 0.466z_1^4z_2^2 \\ & + 0.038z_1^5z_2^2 + 0.367z_2^3 - 1.350z_1z_2^3 - 0.147z_1^2z_2^3 \\ & - 0.170z_1^3z_2^3 + 0.088z_1^4z_2^3 - 0.009z_1^5z_2^3 - 0.205z_2^4 \\ & + 0.223z_1z_2^4 + 0.263z_1^2z_2^4 - 0.185z_1^3z_2^4 + 0.033z_1^4z_2^4 \\ & - 0.001z_1^5z_2^4 + 0.013z_2^5 - 0.000z_1z_2^5 - 0.029z_1^2z_2^5 \\ & + 0.025z_1^3z_2^5 - 0.005z_1^4z_2^5 + 0.000z_1^5z_2^5.\end{aligned} \tag{8.14}$$

Figure 8.8 presents this constitutive constraint, $\hat{g}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour) by the light green over the region formed by the complete sets of feasible (blue) and infeasible (red) points. There is no occasion to judge the success of our algorithm by just comparing the identified constitutive constraint given by Equation (8.14) with the unknown or actual constraints given in Equation (8.6) since we are approximating the feasible region formed by the six constraints via a single 5th-order polynomial constitutive constraint. However, it is possible to observe from Figure 8.8 that the identified constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. The zero values of J_1 and J_2 in Table 8.9 imply this as well.

With the second example of this section, seven inequality constraints which will be treated as unknowns and will be detected by our algorithm had already been presented in Section 8.2 with Equation (8.4). We approximate the seven inequality constraints forming the non-convex region via a single 7th-order polynomial constitutive constraint. Additionally, the boundary-zone formation procedure was also not used. Thus, MS is employed to generate the full set of sample points as outlined in Chapter 3. Here, it should be underlined that, for this example, the CMA-ES (Covariance Matrix Adaptation Evolution Strategy) optimizer was utilized since the DE optimizer was not successful in solving this difficult example.

CMA-ES is a stochastic, derivative-free optimization algorithm that relies on the estimation of a positive definite covariance matrix of decision variables (Hansen et al., 2003).

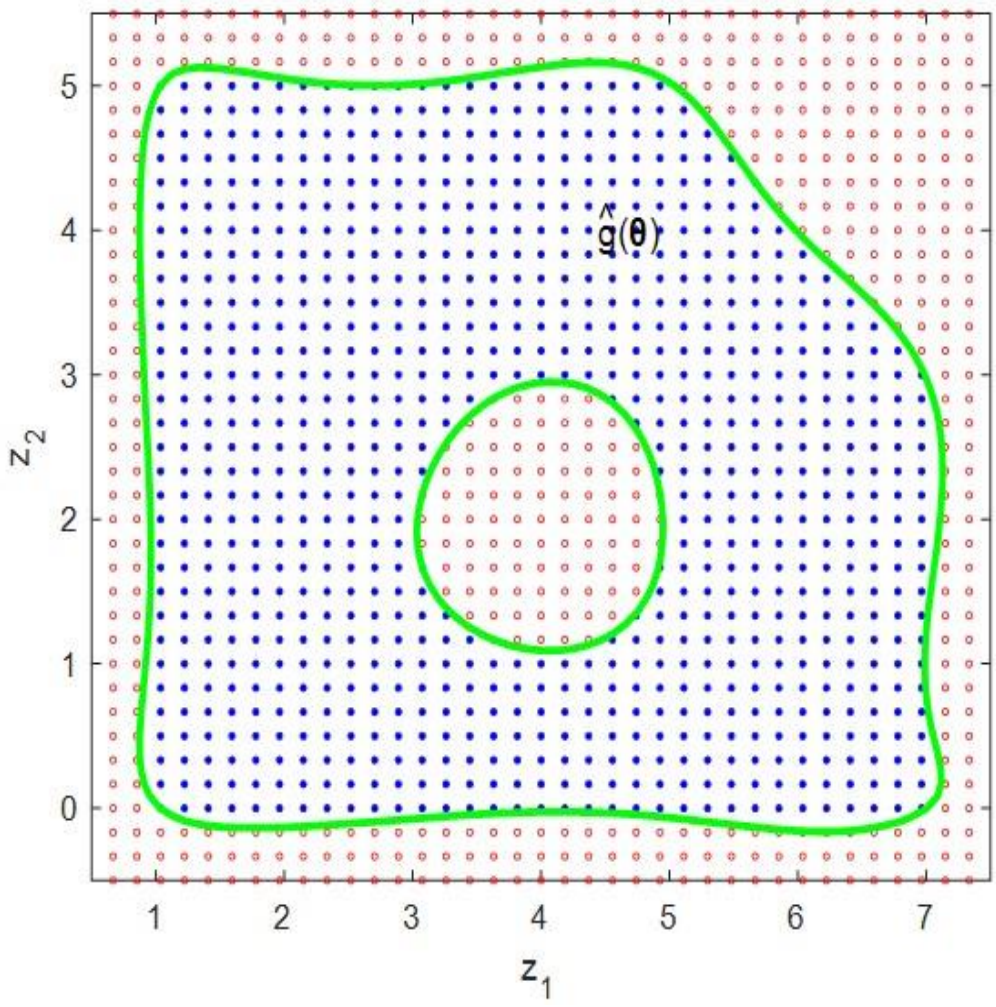

Figure 8.8. Identified optimal single 5th-order polynomial constitutive constraint superimposed with the complete sets of feasible and infeasible points forming a disjoint infeasible region.

Information on the sampling and optimization, as well as the results are all presented in Table 8.10.

The form-free 7th-order polynomial constitutive constraint identified is not given explicitly here since it has $|\boldsymbol{\theta}| = 64$ parameters.

Figure 8.9 presents this constitutive constraint, $\hat{g}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour) by the light green over the region formed by the complete sets of feasible (blue) and infeasible (red) points.

It is possible to observe from Figure 8.9 that the identified constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. The zero values of J_1 and J_2 in Table 8.10 imply this as well.

Table 8.10. Sampling and optimization information and the results.

Sampling Information				
$\mathbf{z}^{LB} = [0 \ -1]$		$\mathbf{z}^{UB} = [+8 \ +6]$		
$N = 1500$		$N^f = 474$		$N^i = 1047$
$\mathbf{Z} \in \mathbb{R}^{1500\times 2}$		$\mathbf{X} \in \mathbb{R}^{474\times 2}$		$\mathbf{Y} \in \mathbb{R}^{1047\times 2}$
Optimization Settings				
$N^{Con} = 1$	$m = 7$	$D = 2$	$\lvert\boldsymbol{\theta}\rvert = 64$	$\mu = 10^{-6}$
$\mathbf{z}^L = [-5 \ -5]$	$\mathbf{z}^U = [+5 \ +5]$		$NP = 150$	$Tol = 10^{-9}$
Optimization Results				
Iterations $= 6667$	Function Evaluations $= 1000051$			
$J = 1.198 \times 10^{-5}$	$J_1 = 0$	$J_2 = 0$	$J_3 = 11.985$	
Perimeter $= 24.524$	Area $= 18.921$			

In this section, we presented two constraint-identification examples of high-order polynomial constitutive constraints. The first one was a 5th-order polynomial constitutive constraint for the approximation of a disjoint infeasible region and had thirty-six parameters. The second was a 7th-order polynomial constitutive constraint for the approximation of a non-convex feasible region and had sixty-four parameters. By these examples, we showed that a single constitutive (single output) but high order model could identify the complex regions formed by several inequality constraints.

Here, the models were polynomials of high order (but they were linear in parameters). This gave us the idea of using nonlinear implicit models such as Neural

Networks (NN) and Extreme Learning Machines (ELM), which are also high order but nonlinear models, as implicit constitutives with probably a single output. We expect that NN / ELM with single output can identify complex regions such as ones analysed in this section.

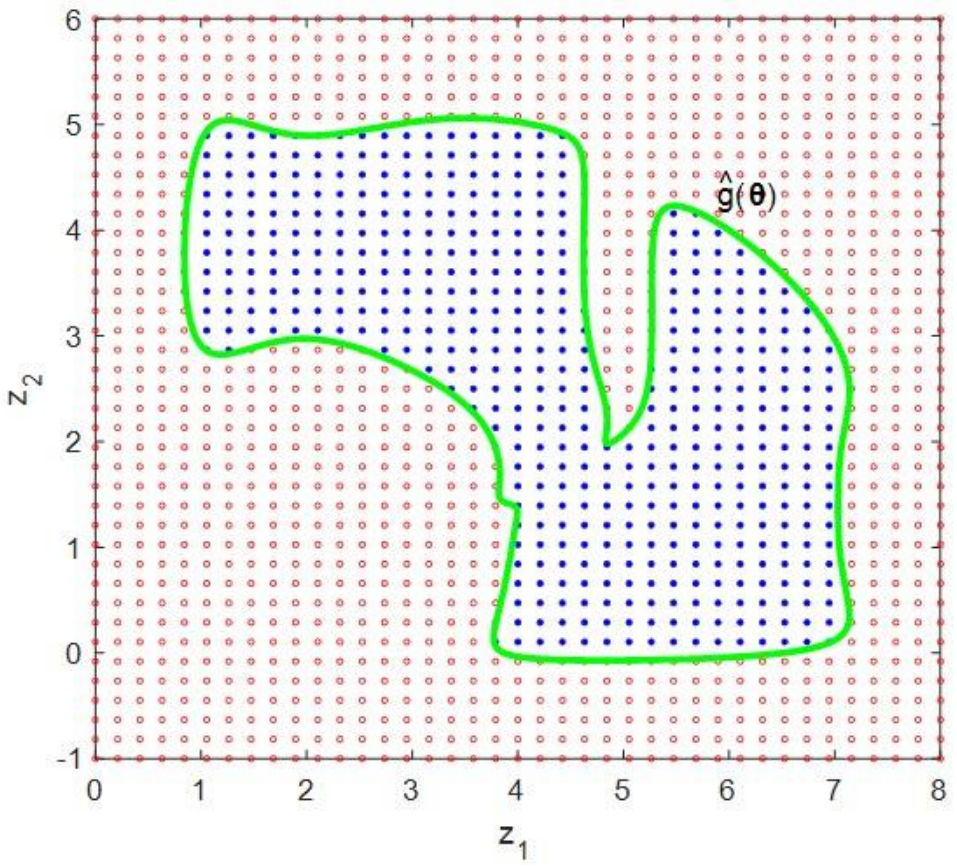

Figure 8.9. Identified optimal single 7th-order polynomial constitutive constraint superimposed with the complete sets of feasible and infeasible points forming a non-convex feasible region.

8.8. A Heart-Shaped Region in 2-D

In this section, our algorithm will approximate a heart-shaped region formed by a complex nonlinear inequality constraint. The inequality constraint, forming the heart-shaped feasible region, which will be treated as unknown and will be detected by our algorithm, is given by the following equation:

$$g_1 : +z_1^2 + \left(1.25z_2 - 2\sqrt{|z_1|}\right)^2 - 5 \leq 0. \tag{8.15}$$

With the first example of this section, we identify such an intricate inequality constraint forming the non-convex feasible region, called “You are in my heart”, since the feasible points are inside the heart region, via a single 4th-order polynomial constitutive constraint. Also, in this part, the boundary-zone formation procedure was not used and SLHS is employed to generate the full set of sample points.

Information on the sampling and optimization, as well as the results are all presented in Table 8.11.

Table 8.11. Sampling and optimization information and the results.

Sampling Information				
$\mathbf{z}^{\mathrm{LB}} = [-3 \;\; -2.5]$	$\mathbf{z}^{\mathrm{UB}} = [+3 \;\; +4]$			
$\mathrm{N} = 2500$	$\mathrm{N}^{\mathrm{f}} = 808$			$\mathrm{N}^{\mathrm{i}} = 1692$
$\mathbf{Z} \in \mathbb{R}^{2500\times 2}$	$\mathbf{X} \in \mathbb{R}^{808\times 2}$			$\mathbf{Y} \in \mathbb{R}^{1692\times 2}$
Optimization Settings				
$\mathrm{N}^{\mathrm{Con}} = 1$	$\mathrm{m} = 4$	$\mathrm{D} = 2$	$\lvert\boldsymbol{\theta}\rvert = 25$	$\mu = 10^{-6}$
$\mathbf{z}^{\mathrm{L}} = [-5 \;\; -5]$	$\mathbf{z}^{\mathrm{U}} = [+5 \;\; +5]$		$\mathrm{NP} = 150$	$\mathrm{Tol} = 10^{-9}$
Optimization Results				
$\mathrm{Iterations} = 88466$	$\mathrm{Function\ Evaluations} = 6414268$			
$\mathrm{J} = 1.964 \times 10^{-6}$	$\mathrm{J}_1 = 0$	$\mathrm{J}_2 = 0$	$\mathrm{J}_3 = 1.964$	
$\mathrm{Perimeter} = 15.486$	$\mathrm{Area} = 12.598$			

The form-free 4th-order polynomial constitutive constraint identified is given explicitly, together with the optimal values of parameters, by the following equation:

$$\begin{aligned}\hat{\mathrm{g}}(\boldsymbol{\theta}) = \; & -0.017 + 0.100\mathrm{z}_1 - 0.310\mathrm{z}_1^2 - 0.098\mathrm{z}_1^3 + 0.329\mathrm{z}_1^4 \\ & + 0.002\mathrm{z}_2 + 0.102\mathrm{z}_1\mathrm{z}_2 - 0.454\mathrm{z}_1^2\mathrm{z}_2 + 0.037\mathrm{z}_1^3\mathrm{z}_2 \\ & - 0.165\mathrm{z}_1^4\mathrm{z}_2 - 0.001\mathrm{z}_2^2 - 0.043\mathrm{z}_1\mathrm{z}_2^2 + 0.082\mathrm{z}_1^2\mathrm{z}_2^2 \\ & + 0.013\mathrm{z}_1^3\mathrm{z}_2^2 + 0.054\mathrm{z}_1^4\mathrm{z}_2^2 - 0.004\mathrm{z}_2^3 - 0.025\mathrm{z}_1\mathrm{z}_2^3 \\ & + 0.070\mathrm{z}_1^2\mathrm{z}_2^3 - 0.004\mathrm{z}_1^3\mathrm{z}_2^3 - 0.015\mathrm{z}_1^4\mathrm{z}_2^3 + 0.002\mathrm{z}_2^4 \\ & + 0.008\mathrm{z}_1\mathrm{z}_2^4 - 0.014\mathrm{z}_1^2\mathrm{z}_2^4 - 0.000\mathrm{z}_1^3\mathrm{z}_2^4 + 0.001\mathrm{z}_1^4\mathrm{z}_2^4.\end{aligned} \tag{8.16}$$

Figure 8.10 presents this constitutive constraint, $\hat{g}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour) by the light green over the region formed by the complete sets of feasible (blue) and infeasible (red) points. There is no occasion to judge the success of our algorithm by just comparing the identified constitutive constraint given by Equation (8.16) with the unknown or actual constraint given in Equation (8.15) since we are approximating a non-polynomial nonlinear feasible region formed by g_1 via a single 4th-order polynomial constitutive constraint. However, it is possible to observe from Figure 8.10 that the identified constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. The zero values of $J_1$and J_2 in Table 8.11 imply this as well.

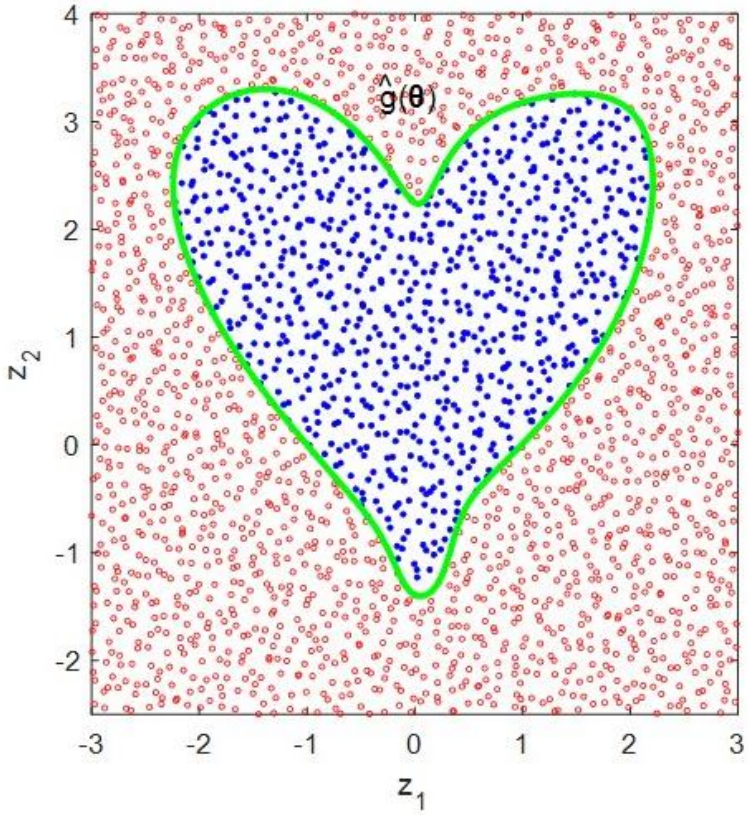

Figure 8.10. Identified optimal single 4th-order polynomial constitutive constraint superimposed with the complete sets of feasible and infeasible points forming "you are in my heart" version of the heart-shaped region.

With the second example of this section, we identify the same intricate nonlinear inequality constraint (Equation (8.15)) by interchanging the feasible and infeasible zones

through multiplying the constraint function with minus one, i.e., $-g_1$, to form the non-convex feasible region called "You are not in my heart" since the feasible points are outside the heart region, via a single 4th-order polynomial constitutive constraint.

Information on the sampling and optimization, as well as the results are all presented in Table 8.12.

Table 8.12. Sampling and optimization information and the results.

Sampling Information				
$\mathbf{z}^{\mathrm{LB}} = [-3 \ -2.5]$	$\mathbf{z}^{\mathrm{UB}} = [+3 \ +4]$			
$\mathrm{N} = 2500$	$\mathrm{N}^{\mathrm{f}} = 1692$			$\mathrm{N}^{\mathrm{i}} = 808$
$\mathbf{Z} \in \mathbb{R}^{2500\times 2}$	$\mathbf{X} \in \mathbb{R}^{1692\times 2}$			$\mathbf{Y} \in \mathbb{R}^{808\times 2}$
Optimization Settings				
$\mathrm{N}^{\mathrm{Con}} = 1$	$\mathrm{m} = 4$	$\mathrm{D} = 2$	$\lvert\boldsymbol{\theta}\rvert = 25$	$\mu = 10^{-6}$
$\mathbf{z}^{\mathrm{L}} = [-5 \ -5]$	$\mathbf{z}^{\mathrm{U}} = [+5 \ +5]$		$\mathrm{NP} = 150$	$\mathrm{Tol} = 10^{-9}$
Optimization Results				
$\mathrm{Iterations} = 156828$	$\mathrm{Function\ Evaluations} = 16313253$			
$\mathrm{J} = 1.194 \times 10^{-8}$	$\mathrm{J}_1 = 0$	$\mathrm{J}_2 = 0$	$\mathrm{J}_3 = 0.011$	
$\mathrm{Perimeter} = 15.843$	$\mathrm{Area} = 12.598$			

The form-free 4th-order polynomial constitutive constraint identified is given explicitly, together with the optimal values of parameters, by the following equation:

$$\begin{aligned}\hat{g}(\boldsymbol{\theta}) = \ & 10^{-4}(+0.663 - 0.000z_1 + 25.401z_1^2 - 0.000z_1^3 \\ & - 25.496z_1^4 - 0.740z_2 - 0.000z_1z_2 + 25.611z_1^2z_2 \\ & + 0.000z_1^3z_2 + 18.398z_1^4z_2 + 0.050z_2^2 + 0.000z_1z_2^2 \\ & - 14.120z_1^2z_2^2 - 0.001z_1^3z_2^2 - 6.447z_1^4z_2^2 + 0.383z_2^3 \\ & + 0.012z_1z_2^3 + 0.011z_1^2z_2^3 - 0.024z_1^3z_2^3 + 1.400z_1^4z_2^3 \\ & - 0.150z_2^4 - 0.007z_1z_2^4 + 0.406z_1^2z_2^4 + 0.008z_1^3z_2^4 \\ & - 0.139z_1^4z_2^4).\end{aligned} \tag{8.17}$$

Figure 8.11 presents this constitutive constraint, $\hat{g}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour) by the light green over the region formed by the complete sets of feasible (blue) and infeasible (red) points. There is no occasion to judge the success of our algorithm by just comparing the identified constitutive constraint given by Equation (8.17) with the unknown or actual constraint given in Equation (8.15) since we are approximating a non-polynomial nonlinear feasible region formed by $-g_1$ via a single 4th-order polynomial constitutive constraint. However, it is possible to observe from Figure 8.11 that the identified constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. The zero values of $J_1$and J_2 in Table 8.12 imply this as well.

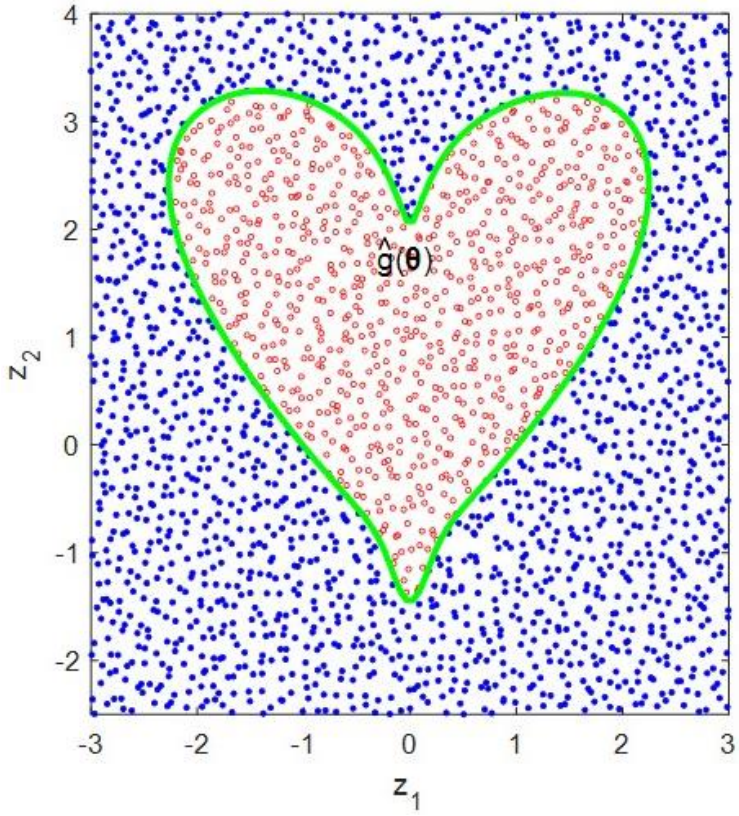

Figure 8.11. Identified optimal single 4th-order polynomial constitutive constraint superimposed with the complete sets of feasible and infeasible points forming "you are not in my heart" version of the heart-shaped region.

Lastly, we can compare single 4th-order polynomial constitutive constraints identified for both examples. The border function, $g_1 = 0$, between the feasible and infeasible

regions in both versions is identical. The region of the second example is obtained by multiplying the constraint separating the regions by minus one. Thus, we expect the signs of the coefficients of the identified constraints to have opposite signs, and indeed, the coefficients in Equation (8.16) and Equation (8.17) have the opposite signs However, the magnitudes of the coefficients of both cases are, in general, not similar. However, we know, and it had been discussed in Chapter 5, that the coefficients of the identified constraints may not be unique. Thus, it is possible that two different constitutive constraints indicate the same feasible or infeasible region (same boundary).

8.9. Octagonal Approximation of Convex Circular Region

In the last section of Chapter 8, we will provide an example for the approximate identification of a convex-circular region by an octagon generated with the form-specific linear and bound constitutive constraints. Unlike previous sections of this chapter, we will not present the detailed information on sampling, optimization, and the detailed results of optimization so as to keep the focus only on the octagonal (piecewise linear) approximate identification.

The inequality constraint, forming the convex-circular region, which will be treated as unknown and will be approximated by our algorithm in the form of an octagon, is given by the following equation:

$$g_1: \quad (z_1 - 0.5)^2 + (z_2 - 0.5)^2 + 0.0625 \leq 0. \tag{8.18}$$

The SLHS method is used to generate sample points with $N = 2500$. The procedure of the formation of the boundary zone is applied to this example via $p = 3$ (Chapter 3). The value of rho is set to $\rho = 500$ for the KS function that aggregates the form-specific constitutive constraints (Chapter 4).

To approximately identify the convex-circular boundary as octagonal, we will employ constitutive bound constraints (see Equation (7.6) in Chapter 7) together with four constitutive linear constraints (see Equation (7.7) in Chapter 7). Thus, the number of the optimization decision variables becomes $|\boldsymbol{\theta}| = 16$. To solve this example, the CMA-ES

optimizer is used. The norm of the parameters is included in the objective function with $\mu = 10^{-6}$.

The form-specific constitutive bound constraints and constitutive linear inequality constraints identified are given explicitly by:

$$\hat{g}_1\left(\boldsymbol{\theta}_1^{\mathrm{UB}}\right) = -0.739 + z_1, \tag{8.19a}$$

$$\hat{g}_2\left(\boldsymbol{\theta}_2^{\mathrm{UB}}\right) = -0.741 + z_2, \tag{8.19b}$$

$$\hat{g}_3\left(\boldsymbol{\theta}_1^{\mathrm{LB}}\right) = +0.255 - z_1, \tag{8.19c}$$

$$\hat{g}_4\left(\boldsymbol{\theta}_2^{\mathrm{LB}}\right) = +0.258 - z_2, \tag{8.19d}$$

$$\hat{g}_5\left(\boldsymbol{\theta}_1^{\mathrm{L}}\right) = +0.931 - 1.386z_1 - 1.415z_2, \tag{8.19e}$$

$$\hat{g}_6\left(\boldsymbol{\theta}_2^{\mathrm{L}}\right) = -0.402 + 0.259z_1 + 0.340z_2, \tag{8.19f}$$

$$\hat{g}_7\left(\boldsymbol{\theta}_3^{\mathrm{L}}\right) = -0.084 + 0.235z_1 - 0.231z_2, \tag{8.19g}$$

$$\hat{g}_8\left(\boldsymbol{\theta}_4^{\mathrm{L}}\right) = -0.161 - 0.362z_1 + 0.413z_2. \tag{8.19h}$$

Figure 8.12 presents the octagonal-shaped aggregated constraint, $\mathbb{C}(\mathbf{P})$, by the light green and the identified individual constitutive constraints by the claret red, purple, cyan, yellow lines (constitutive bound constraints), and by the orange, dark blue, black, dark green lines (constitutive linear inequality constraints) over the region formed by the boundary sets of feasible (blue) and infeasible (red) points. It is possible to observe from Figure 8.12 that the aggregated constraint leads to some violations at the boundary. We obtained the results of the optimization as $J_1 = 0.030$ and $J_2 = 8$, with 5 feasible points out of 211 and 3 infeasible points out of 301. These values also imply the existence of violation of some points. Thus, we obtained the approximate octagonal representation of the convex-circular region.

As can be observed from the below figure, our algorithm achieved approximate, but perfect, octagonal representation of the circular boundary by the form-specific constitutive bound constraints and constitutive linear inequality constraints. The deployment of the simple constitutive bound constraints allowed us to obtain the perfect octagonal representation (zero slopes with respect to z_1 on the top and bottom, and zero slopes with respect to z_2 on the left and right).

We can accomplish such piecewise linear approximation for any convex region. However, such linear approximations are impossible for the nonconvex regions as explained in Chapter 2.

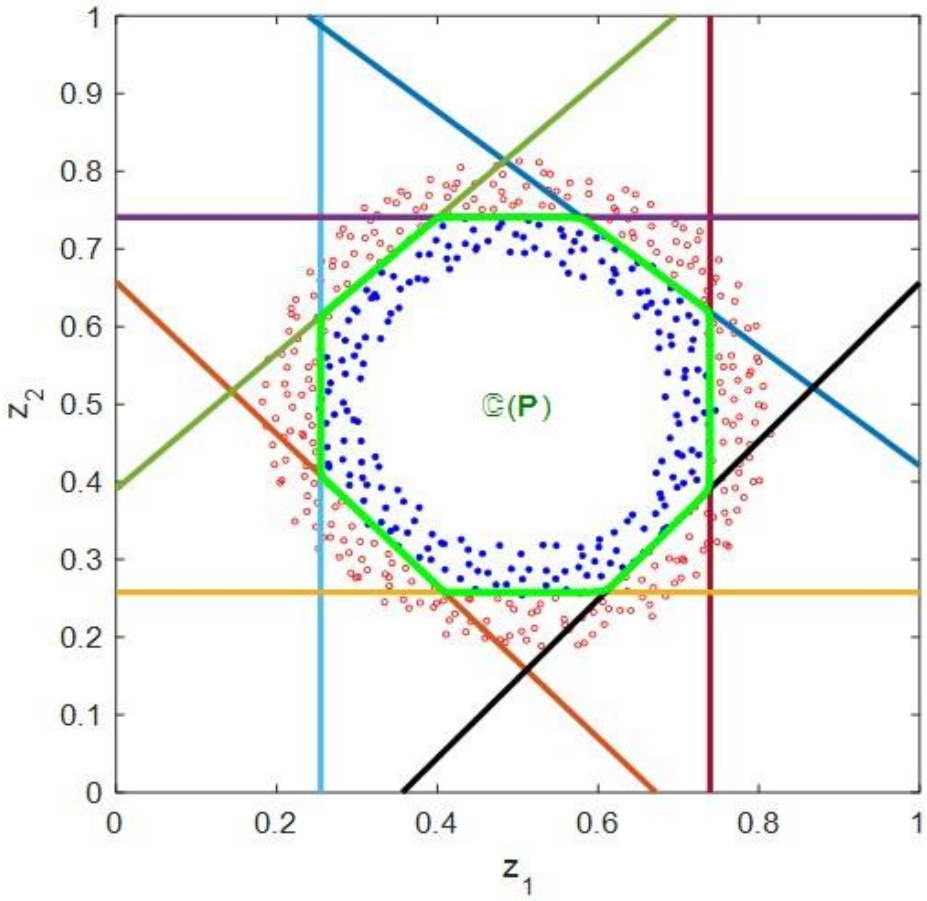

Figure 8.12. Identified constitutive constraints and aggregated constraint representing approximate octagonal boundary superimposed with the boundary sets of feasible and infeasible points forming convex-circular region.

9. CONSTRAINT IDENTIFICATION VIA MACHINE LEARNING

In this chapter, we will equip our algorithm with some Machine Learning (ML) tools such as Neural Networks (NNs) and Extreme Learning Machines (ELMs). This chapter will begin with a short explanation of these machine learning tools and continue with how our algorithm will deploy these ML tools for constraint-identification task. Lastly, we will complete this chapter with some constraint-identification examples solved via these ML tools.

Our algorithm was equipped with the design matrix approach to create models of form-specific and form-free explicit constitutive constraints in Chapter 7. Our algorithm could form multiple lower-order polynomial constitutive constraints or a single higher-order polynomial constitutive constraint to approximate the feasible and infeasible regions. As can be remembered, our algorithm took advantage of the KS aggregation function in order to reduce multiple of such constitutive constraints to a single aggregated constitutive constraint. However, our algorithm did not require the KS aggregation function when utilizing a single (possibly higher-order) polynomial constitutive constraint. Additionally, our algorithm could give the explicit mathematical form of the optimal constitutive constraint equations via the design matrix approach. All constitutive constraints, up until now, were formulated as ‘linear in parameters’ models. In this chapter, instead of the deployment of such explicit models for the creation of the constitutive constraints, our algorithm will firstly employ NNs, as ‘non-linear in parameters’ implicit models, which will directly produce the form-free constitutive constraint(s) without the creation of explicit model functions. The reason why we bring NNs or ELMs into the picture is to capitalize on the nonlinearity inherent in activation functions of these tools and their capacities for approximation of highly nonlinear functions, thus, to easily approximate even the most complex feasible and infeasible regions.

NNs are also known for their universal approximation capabilities, and they are widely utilized for several purposes such as function approximation, feature selection, and pattern recognition in various domains (Han et al., 2019). There are many types of NNs such as Single Layer Feedforward Neural Networks (SLFNNs), multiple layer feedforward

NNs, convolutional NNs (CNN), recurrent NNs and generative adversarial networks (GAN). Our algorithm will only employ SLFNN among these algorithms in order to learn (actually, to identify) unknown inequality constraints via the form-free implicit "neural constitutive constraint(s)". SLFNNs consist of an input layer, a single hidden layer and an output layer. Schematic representation of the SLFNN, including $\mathrm{N}^{\mathcal{H}}$ nodes in the hidden layer and $\mathrm{N}^{\mathcal{O}}$ nodes in the output layer, is given in Figure 9.1.

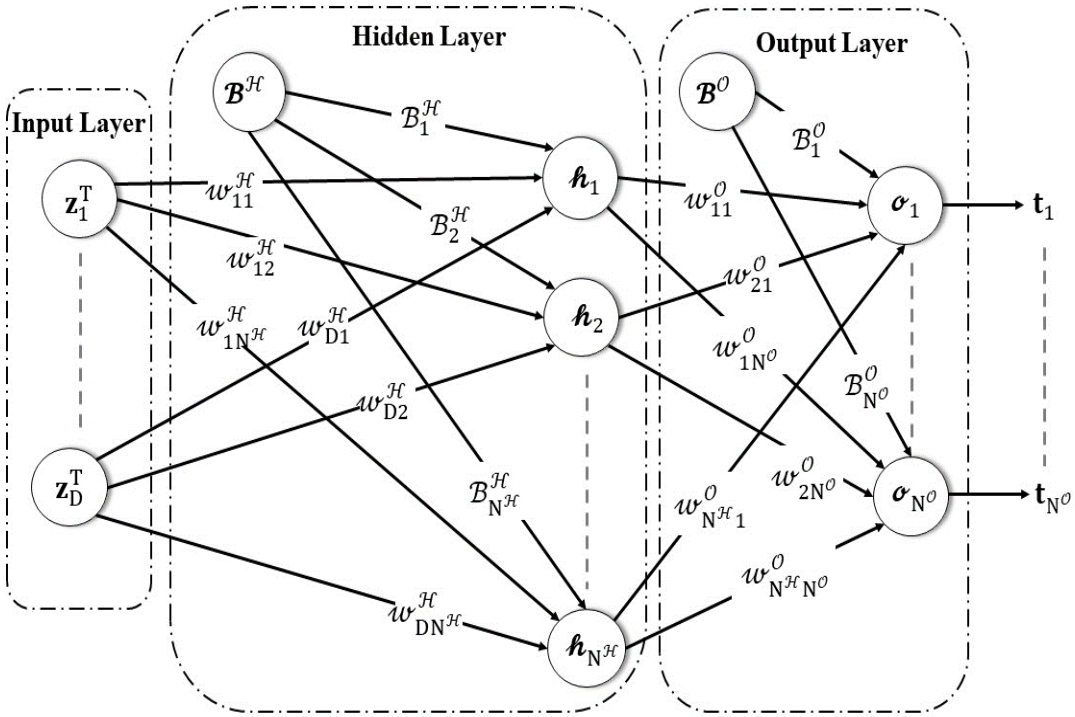

Figure 9.1. The schematic representation of the SLFNN.

The weight vector of hidden layer, connecting i^{th} input neuron to all hidden neurons, becomes $\boldsymbol{w}_{\mathrm{i}}^{\mathcal{H}} = \left[w_{\mathrm{i}1}^{\mathcal{H}} \quad \cdots \quad w_{\mathrm{i}\mathrm{N}^{\mathcal{H}}}^{\mathcal{H}}\right]^{\mathrm{T}} \in \mathbb{R}^{\mathrm{N}^{\mathcal{H}}}$ for $\mathrm{i} = \{1, \dots, \mathrm{D}\}$, the weight vector of output layer, connecting m^{th} hidden neuron to all output neurons, becomes $\boldsymbol{w}_{\mathrm{m}}^{\mathcal{O}} = \left[w_{\mathrm{m}1}^{\mathcal{O}} \quad \cdots \quad w_{\mathrm{m}\mathrm{N}^{\mathcal{O}}}^{\mathcal{O}}\right]^{\mathrm{T}} \in \mathbb{R}^{\mathrm{N}^{\mathcal{O}}}$ for $\mathrm{m} = \{1, \dots, \mathrm{N}^{\mathcal{H}}\}$ and $\boldsymbol{\mathcal{B}}^{\mathcal{H}} \in \mathbb{R}^{\mathrm{N}^{\mathcal{H}}}$ and $\boldsymbol{\mathcal{B}}^{\mathcal{O}} \in \mathbb{R}^{\mathrm{N}^{\mathcal{O}}}$ designate the bias vectors of hidden and output layers, respectively. The outputs of the nodes in the hidden layer are computed as $\boldsymbol{h}_{\mathrm{m}} = \varphi\left(\mathcal{B}_1^{\mathcal{H}} + w_{1\mathrm{m}}^{\mathcal{H}}\mathbf{z}_1^{\mathrm{T}} + \cdots + w_{\mathrm{D}\mathrm{m}}^{\mathcal{H}}\mathbf{z}_{\mathrm{D}}^{\mathrm{T}}\right)$ for $\mathrm{m} = \{1, \dots, \mathrm{N}^{\mathcal{H}}\}$ and $\varphi(.)$ denotes any activation function. The outputs of the nodes in the output layer are also computed in the same way, but the outputs of the nodes in the hidden layer become the inputs of the nodes in the output layer.

The values obtained by the multiplication and summation operators goes through an activation function before regarded as the outputs of the nodes in the layer. Activation function enables a node in any layer to map its input values into its output values of certain ranges. It also provides the NN its nonlinearity. For instance, the sigmoid activation function, commonly used in the hidden layers, maps its inputs to its outputs between zero and one, while the hyperbolic tangent function maps its inputs to its outputs between minus one and plus one, both of which are nonlinear activation functions. Radial basis function (Gaussian function) is generally utilized as the activation function in the single hidden layer of radial basis function networks and maps its inputs to its outputs between zero and one. Hyperbolic secant activation function is among the nonlinear activation functions that return their output values as between zero and one. The outputs of the nodes in a layer become the values obtained only by multiplication and summation operator when linear activation function, called identity function, is utilized. Identity function is generally used in the nodes of the output layer. Rectified Linear Unit (ReLU) is nonlinear but also non-smooth activation function which makes a negative input value zero and keeps a positive value as it is. Softmax activation function is chosen in the nodes of the output layer for multi-class classification problems (Samarasinghe, 2006; Samatin Njikam, and Zhao, 2016). These activation functions are not explicitly given here, but they will be given with the constraint-identification examples when utilized.

On the other hand, Random Single Layer Feedforward Neural Networks (RSLFNNs), also more widely called Extreme Learning Machines (ELMs), has been introduced to expedite the training process of SLFNNs firstly by setting the weight and bias terms, connecting the input layer to the hidden layer, to randomly assigned numbers, secondly, not necessarily but usually, by discarding the bias terms of output layer of the SLFNN, and thus by eliminating the training (optimization) phase completely. Additionally, linear activation functions are utilized in the output layer of the ELMs. Thus, the weight terms of the output layer of ELM can be estimated --without a training phase-- via pseudo-inverse operation of the output matrix of the hidden layer given the output (target) data (as numbers or as discrete labels) for time-series regression and forecasting applications or for classification tasks. In other words, with these ELM settings (i.e., by setting the weight and bias terms of the hidden layer to the randomly assigned values and by utilizing linear activation functions in the output layer), the training process of ELMs

with one hidden layer reduces to the solution of a under-determined linear system of algebraic equations (G.-B. Huang et al., 2004). However, in this thesis work, we do not have a direct output (target) data to fit via ELM's input data, and thus, we cannot capitalize on the advantage of ELMs being training-free due to sufficiency of using pure matrix algebra. Salient property of ELM is that they prove the same approximation capabilities as SLFNNs with much less parameters and without a training phase. Thus, our algorithm, equipped with ELM, will be able to take advantage of the approximation capability of SLFNNs while significantly reducing the number of the optimization decision variables (which are only the output weights and --if utilized-- output biasses, since the hidden layer weights / biasses are randomly assigned). Our algorithm equipped with ELM directly generate the form-free constitutive constraint(s) without the need for any explicit model function. However, as opposed to the constitutive constraints generated via SLFNN algorithm (will be referred to simply as "NN" after this point onwards), they become linear in parameters. Schematic representation of the ELMs thus may also be given with Figure 9.1 with the replacement of hidden layer weights / biasses by random numbers and --if the output biasses are not used-- with setting the output biasses to zero.

Before giving how our algorithm generates the form-free implicit constitutive constraints and executes the task of the "constraint learning" (actually, still "constraint identification") via these ML tools, it should be highlighted that the form-free implicit constitutive constraints, when evaluated at the sets of feasible and infeasible points, are treated as mentioned in Chapter 5 and Chapter 7.

By the deployment of the NN, the form-free implicit constitutive constraint(s) are formed as in the following equation:

$$\begin{bmatrix} \hat{\mathbf{g}}_1^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}) \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \vdots \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \hat{\mathbf{g}}_{\mathrm{N}^{\mathcal{O}}}^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}) \end{bmatrix} \coloneqq \mathcal{N}_{\mathrm{NN}}(\mathbf{Z},\boldsymbol{\theta}) = \varphi_{\mathcal{O}}\left(\boldsymbol{\mathcal{W}}^{\mathcal{O}}\left(\varphi_{\mathcal{H}}(\boldsymbol{\mathcal{W}}^{\mathcal{H}}\mathbf{Z}^{\mathrm{T}} + \boldsymbol{\mathcal{B}}^{\mathcal{H}})\right) + \boldsymbol{\mathcal{B}}^{\mathcal{O}}\right), \qquad (9.1)$$

where $\boldsymbol{\mathcal{W}}^{\mathcal{H}} = [\boldsymbol{w}_1^{\mathcal{H}} \quad \cdots \quad \boldsymbol{w}_{\mathrm{D}}^{\mathcal{H}}] \in \mathbb{R}^{\mathrm{N}^{\mathcal{H}}\times\mathrm{D}}$ and $\boldsymbol{\mathcal{B}}^{\mathcal{H}} = \left[\mathcal{B}_1^{\mathcal{H}} \quad \cdots \quad \mathcal{B}_{\mathrm{N}^{\mathcal{H}}}^{\mathcal{H}}\right]^{\mathrm{T}} \in \mathbb{R}^{\mathrm{N}^{\mathcal{H}}}$ denotes the weight matrix and the bias vector of the hidden layer, $\boldsymbol{\mathcal{W}}^{\mathcal{O}} = \left[\boldsymbol{w}_1^{\mathcal{O}} \quad \cdots \quad \boldsymbol{w}_{\mathrm{N}^{\mathcal{H}}}^{\mathcal{O}}\right] \in \mathbb{R}^{\mathrm{N}^{\mathcal{O}}\times\mathrm{N}^{\mathcal{H}}}$ and $\boldsymbol{\mathcal{B}}^{\mathcal{O}} = \left[\mathcal{B}_1^{\mathcal{O}} \quad \cdots \quad \mathcal{B}_{\mathrm{N}^{\mathcal{O}}}^{\mathcal{O}}\right]^{\mathrm{T}} \in \mathbb{R}^{\mathrm{N}^{\mathcal{O}}}$ implies the weight matrix and bias vector of

the output layer, and $\mathcal{N}_{\mathrm{NN}}(.)$ represent the NN function and its outcome (output(s)). The symbol of φ also denotes any activation function as mentioned above. Additionally, the parameters of the constitutive constraint(s) become $\boldsymbol{\theta} = [(\mathrm{vec}(\boldsymbol{\mathcal{W}}^{\mathcal{H}}))^{\mathrm{T}} \quad (\boldsymbol{\mathcal{B}}^{\mathcal{H}})^{\mathrm{T}} \quad (\mathrm{vec}(\boldsymbol{\mathcal{W}}^{\mathcal{O}}))^{\mathrm{T}} \quad (\boldsymbol{\mathcal{B}}^{\mathcal{O}})^{\mathrm{T}}]^{\mathrm{T}}$, where $\mathrm{vec}(\mathbf{A}) \in \mathbb{R}^{\mathrm{mn}}$ stands for the vectorization of the matrix $\mathbf{A} \in \mathbb{R}^{\mathrm{m}\times\mathrm{n}}$ and is the column vector obtained by stacking the columns of the matrix $\mathbf{A}$ on top of one another. $\mathrm{N}^{\mathcal{O}} = 1$ indicates that there is only one output to represent the single (unaggregated) constitutive constraint, and $\mathrm{N}^{\mathcal{O}} > 1$ indicates that there are more than one output to represent the constitutive constraints that will be aggregated via KS function to identify the feasible and infeasible regions.

By the deployment of the ELM, the form-free implicit constitutive constraint(s) --if the output biasses are not used-- are generated as in the following equation:

$$\begin{bmatrix} \hat{\mathbf{g}}_1^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}) \\ \dots\dots\dots\dots\dots \\ \vdots \\ \dots\dots\dots\dots\dots \\ \hat{\mathbf{g}}_{\mathrm{N}^{\mathcal{O}}}^{\mathrm{T}}(\mathbf{Z},\boldsymbol{\theta}) \end{bmatrix} \coloneqq \mathcal{N}_{\mathrm{ELM}}(\mathbf{Z},\boldsymbol{\theta}) = \boldsymbol{\mathcal{W}}^{\mathcal{O}}\varphi(\boldsymbol{\mathcal{W}}^{\mathcal{H}}\mathbf{Z}^{\mathrm{T}} + \boldsymbol{\mathcal{B}}^{\mathcal{H}}), \tag{9.2}$$

where $\boldsymbol{\mathcal{W}}^{\mathcal{H}} = [\boldsymbol{w}_1^{\mathcal{H}} \quad \cdots \quad \boldsymbol{w}_{\mathrm{D}}^{\mathcal{H}}] \in \mathbb{R}^{\mathrm{N}^{\mathcal{H}}\times\mathrm{D}}$ and $\boldsymbol{\mathcal{B}}^{\mathcal{H}} = \left[\mathcal{B}_1^{\mathcal{H}} \quad \cdots \quad \mathcal{B}_{\mathrm{N}^{\mathcal{H}}}^{\mathcal{H}}\right]^{\mathrm{T}} \in \mathbb{R}^{\mathrm{N}^{\mathcal{H}}}$ denotes the randomly-assigned weight matrix and the randomly-assigned bias vector of the hidden layer, $\boldsymbol{\mathcal{W}}^{\mathcal{O}} = \left[\boldsymbol{w}_1^{\mathcal{O}} \quad \cdots \quad \boldsymbol{w}_{\mathrm{N}^{\mathcal{H}}}^{\mathcal{O}}\right] \in \mathbb{R}^{\mathrm{N}^{\mathcal{O}}\times\mathrm{N}^{\mathcal{H}}}$ implies the weight matrix of the output layer, and $\mathcal{N}_{\mathrm{ELM}}(.)$ represent the ELM function and its outcome (output(s)). The symbol φ denotes any activation function as mentioned above.

As can be understood from Equation (9.1) and Equation (9.2), the number of the form-free constitutive constraints depends on the number of the nodes in the output layer. For instance, if our algorithm utilizes a single output node, i.e., $\mathrm{N}^{\mathcal{O}} = 1$, it approximates the feasible and infeasible regions via a single form-free implicit constitutive constraint i.e., $\hat{\mathbf{g}}(\mathbf{Z},\boldsymbol{\theta})$, and there is no need for the KS aggregation function. If multiple output nodes are deployed i.e., $\mathrm{N}^{\mathcal{O}} > 1$, our algorithm detects the feasible and infeasible regions with the form-free implicit constitutive constraint formed via the aggregation of the $\mathrm{N}^{\mathcal{O}}$ form-free implicit constitutive constraints, i.e., $\mathbb{C}(\mathbf{Z},\boldsymbol{\theta})$. The schematic representation of the aggregation of the form-free implicit constitutive constraints generated via these ML tools with multiple output nodes is illustrated in Figure 9.2.

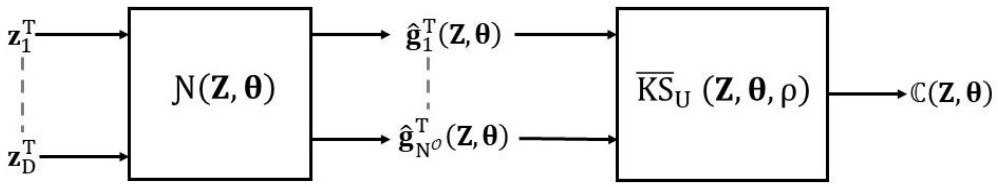

Figure 9.2. The schematic representation of the aggregation of the form-free implicit constitutive constraints obtained via machine learning tools.

To achieve the optimal form-free implicit constitutive constraint(s), our algorithm adjusts the parameters of the NN or ELM (i.e., form-free implicit constitutive constraint(s)) by going through the same optimization procedure, thoroughly discussed in Chapter 5 and Chapter 7, and by utilizing the same objective function (see Equation (5.8) and Equation (7.15)).

Lastly, as can be understood from Equation (9.1) and Equation (9.2), as well as from Figure 9.2, the inputs to NN or ELM comes from the complete sets of the feasible and infeasible points. However, our algorithm feeds NN or ELM with the vectorized mesh grid points obtained based on $\mathbf{z}^{LB}$ and $\mathbf{z}^{UB}$ of the coordinates (which can be regarded as the inputs of the test set) so as to present the zero-valued contours of the optimal (aggregated) constitutive constraint after attaining optimal solution. Actually, this procedure was also employed in Chapter 5 and Chapter 7.

9.1. Learning a Polygonal Convex Feasible Region via Single-Output NN with ReLU Activation Functions

In this section, we will demonstrate our algorithm's ability to detect a polygonal convex feasible region formed by four bound constraints and one linear inequality constraint by employing a NN with single output and equipped with ReLU activation functions.

Five inequality constraints which will be treated as unknowns and will be detected by our NN-based algorithm are given by the following equations:

$$g_1: \quad +z_1 - 7 \leq 0, \tag{9.3a}$$

$$g_2: \quad -z_1 + 1 \leq 0, \tag{9.3b}$$

$$g_3: \quad +z_2 - 5 \leq 0, \tag{9.3c}$$

$$g_4: \quad -z_2 \leq 0, \tag{9.3d}$$

$$g_5: \quad +z_1 + z_2 - 10 \leq 0. \tag{9.3e}$$

NN topology used for this two-dimensional example includes thirty nodes (neurons) in the hidden layer and single neuron in the output layer, i.e., $N^{\mathcal{H}} = 30$ and $N^{\mathcal{O}} = 1$, and thus, the number of optimization decision variables becomes $|\boldsymbol{\theta}| = 121$. Additionally, for convex feasible regions formed by linear inequality constraints (i.e., polygonal regions in 2-D and polyhedral regions in n-D), we propose the utilization of ReLU activation functions in the hidden layer neurons, since it has been proven that NNs with ReLU activation functions in the hidden layer(s) are capable of partitioning n-dimensional space into the linear (polyhedral) regions (C. Huang, 2020). We will use, as is customary, linear activation function in the output node (single output attached to a single neuron with linear activation function, see Figure 9.1). These activation functions are explicitly given for a scalar input variable in the following equation:

$$\text{ReLU Activation Function} : \quad \varphi_{\mathcal{H}}(z) = \max\{0, z\}, \tag{9.4a}$$

$$\text{Linear Activation Function} : \quad \varphi_{\mathcal{O}}(z) = z. \tag{9.4b}$$

To solve this example, we use CMA-ES optimizer as briefly explained with the second example of Section 8.7. The lower and upper bounds on the optimization decision variables (NN weights and biasses) will not be imposed.

SLHS is employed to generate the sample points. With this example, we will illustrate the utilization of different percentile values for feasible and infeasible points ($p_{\mathbf{X}}$ and $p_{\mathbf{Y}}$) for the formation of the boundary sets of feasible and infeasible points. In addition to this, some infeasible points, randomly selected from the complete set of infeasible points, are also added to augment the boundary set of the infeasible points after the creation of the boundary sets in order to prevent probable generation of artificial zero-valued contours within the infeasible region. In other words, we form a boundary zone of

feasible (blue) points, a boundary zone of infeasible (red) points, and, outside the infeasible boundary zone, an additional zone of scattered infeasible points. Information on the sampling, NN settings, and optimization, as well as the results are all presented in Table 9.1.

Table 9.1. Sampling, NN settings, optimization information and the results.

Sampling Information			
$\mathbf{z}^{\mathrm{LB}} = [0 \ -1]$		$\mathbf{z}^{\mathrm{UB}} = [+8 \ \ +6]$	
$\mathrm{N} = 2000$		$\mathrm{N}^{\mathrm{f}} = 1016$	$\mathrm{N}^{\mathrm{i}} = 1009$
$\mathbf{Z} \in \mathbb{R}^{2000\times 2}$		$\mathbf{X} \in \mathbb{R}^{1016\times 2}$	$\mathbf{Y} \in \mathbb{R}^{1009\times 2}$
		$\mathrm{p}_{\mathbf{X}} = 1$	$\mathrm{p}_{\mathbf{Y}} = 0.5$
$\mathrm{N}^{\mathrm{b}} = 691$		$\mathrm{N}^{\mathrm{fb}} = 293$	$\mathrm{N}^{\mathrm{ib}} = 398$
$\mathbf{Z} \in \mathbb{R}^{691\times 2}$		$\mathbf{X}^{\mathrm{b}} \in \mathbb{R}^{293\times 2}$	$\mathbf{Y}^{\mathrm{b}} \in \mathbb{R}^{398\times 2}$
NN and Optimization Settings			
$\mathrm{D} = 2$	$\mathrm{N}^{\mathcal{H}} = 30$	$\mathrm{N}^{\mathrm{Con}} = \mathrm{N}^{\mathcal{O}} = 1$	$\lvert\boldsymbol{\theta}\rvert = 121$
$\mathrm{NP} = 50$		$\mathrm{Tol} = 10^{-9}$	$\mu = 10^{-8}$
Optimization Results			
$\mathrm{Iterations} = 5000$		$\mathrm{Function\ Evaluations} = 250001$	
$\mathrm{J} = 24.319 \times 10^{-8}$		$\mathrm{J}_1 = 0$	$\mathrm{J}_2 = 0$ $\mathrm{J}_3 = 24.319$
$\mathrm{Perimeter} = 20.796$		$\mathrm{Area} = 28.036$	

Figure 9.3 presents the form-free implicit constitutive constraint, $\hat{\mathrm{g}}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour), identified with the NN, by the light green over the region formed by the boundary sets of feasible (blue) and infeasible (red) points.

It is possible to observe from Figure 9.3 that the identified neural constitutive constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points included in the boundary sets. The zero values of $\mathrm{J}_1$and J_2 in Table 9.1 imply this as well.

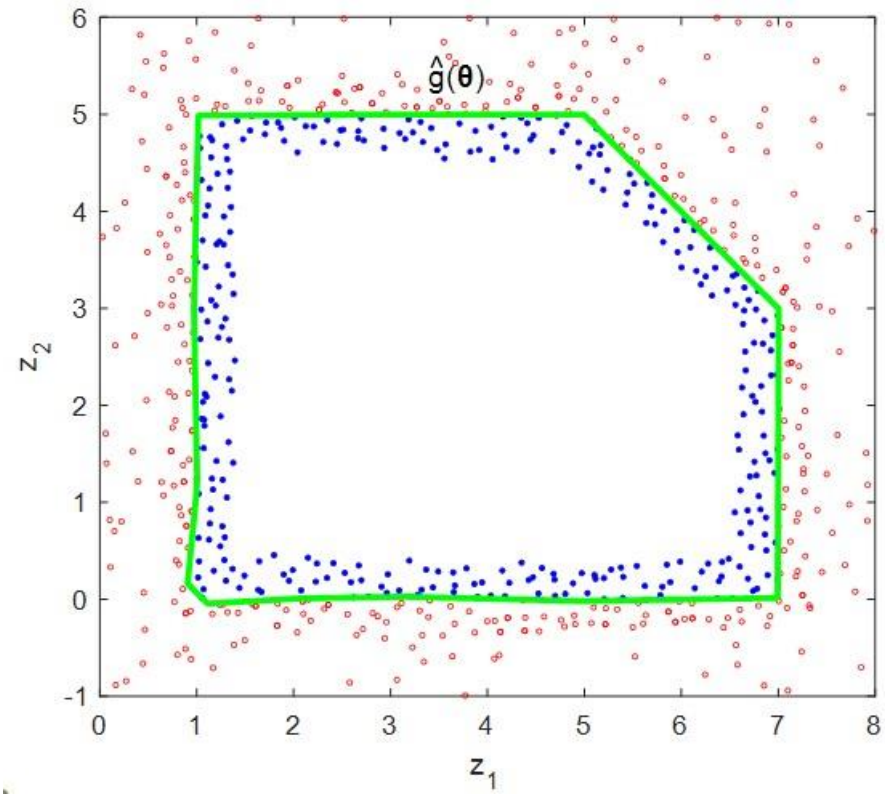

Figure 9.3. Optimal single neural constitutive constraint identified via ReLU activation function, superimposed with the boundary sets of feasible and infeasible points forming a polygonal convex feasible region.

As can be observed from Figure 9.3, with the utilization of ReLU activation function in the hidden layer, our algorithm successfully approximates the convex feasible region formed by linear inequality constraints. In other words, ReLU activation function forces our algorithm to detect the convex feasible region with a neural constitutive constraint forming a piece-wise linear curve, as elaborated by Huang (2020).

It should be mentioned that, although we presented the results obtained with 30 neurons in the hidden layer, it is possible to identify the region almost equally well with much less neurons such as five (when the weight of the third sub-objective function is set to 10^{-2} and the population size is set to 50) in which case the number of optimization decision variables becomes $|\boldsymbol{\theta}| = 21$. However, in this thesis work, we are not concerned with hyperparameter tuning of the ML tools, i.e., the detailed selection of the best number of hidden layers, the best activation function type, and the best number of outputs (the number of output neurons), which are all outside the scope of this thesis.

9.2. Learning a Polygonal Convex Feasible Region via Single-Output NN with Rectified Hyperbolic Secant (ReSech) Activation Functions

In this section, we will demonstrate our algorithm's ability to detect the polygonal convex feasible region, introduced in the previous section, by employing a NN with single output and equipped with Rectified Hyperbolic Secant (ReSech) activation functions.

Five inequality constraints which will be treated as unknowns and will be detected by our NN-based algorithm had been given in Equation (9.3).

NN topology used for this two-dimensional example includes twenty nodes (neurons) in the hidden layer and single neuron in the output layer, i.e., $\mathrm{N}^{\mathcal{H}} = 20$ and $\mathrm{N}^{\mathcal{O}} = 1$, and thus, the number of optimization decision variables becomes $|\boldsymbol{\theta}| = 81$.

To further demonstrate the effectiveness of the utilization of ReLU activation function for such polygonal convex regions, we will test the use of ReSech activation function, introduced by Njikam and Zhao (2016), in the hidden-layer neurons. We will use linear activation function in the output node as usual. The ReSech activation function is explicitly given for a scalar input variable in the following equation:

$$\text{ReSech Activation Function} \ : \quad \varphi_{\mathcal{H}}(\mathrm{z}) = \mathrm{z}\,\mathrm{sech}\,\mathrm{z} = \frac{\mathrm{z}}{\cosh \mathrm{z}}. \tag{9.5}$$

We use CMA-ES optimizer to solve this example. The lower and upper bounds on the optimization decision variables (NN weights and biasses) will not be imposed.

SLHS is employed to generate the sample points as elucidated in Section 9.1. Information on the sampling, NN settings, and optimization, as well as the results are all presented in Table 9.2.

Figure 9.4 presents the form-free implicit constitutive constraint, $\hat{\mathrm{g}}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour), identified with the NN, by the light green over the region formed by the boundary sets of feasible (blue) and infeasible (red) points. It is possible to observe from Figure 9.4 that the identified neural constitutive constraint does not lead to any

violations at the boundary and thus satisfies all the feasible and infeasible points included in the boundary sets. The zero values of $J_1$and J_2 in Table 9.2 imply this as well.

Table 9.2. Sampling, NN settings, optimization information and the results.

Sampling Information			
$\mathbf{z}^{\mathrm{LB}} = [0 \ -1]$		$\mathbf{z}^{\mathrm{UB}} = [+8 \ \ +6]$	
$\mathrm{N} = 2000$		$\mathrm{N}^{\mathrm{f}} = 1013$	$\mathrm{N}^{\mathrm{i}} = 1012$
$\mathbf{Z} \in \mathbb{R}^{2000\times 2}$		$\mathbf{X} \in \mathbb{R}^{1013\times 2}$	$\mathbf{Y} \in \mathbb{R}^{1012\times 2}$
		$\mathrm{p}_{\mathbf{X}} = 1$	$\mathrm{p}_{\mathbf{Y}} = 0.5$
$\mathrm{N}^{\mathrm{b}} = 861$		$\mathrm{N}^{\mathrm{fb}} = 293$	$\mathrm{N}^{\mathrm{ib}} = 568$
$\mathbf{Z} \in \mathbb{R}^{861\times 2}$		$\mathbf{X}^{\mathrm{b}} \in \mathbb{R}^{293\times 2}$	$\mathbf{Y}^{\mathrm{b}} \in \mathbb{R}^{568\times 2}$
NN and Optimization Settings			
$\mathrm{D} = 2$	$\mathrm{N}^{\mathcal{H}} = 20$	$\mathrm{N}^{\mathrm{Con}} = \mathrm{N}^{\mathcal{O}} = 1$	$\lvert\boldsymbol{\theta}\rvert = 81$
$\mathrm{NP} = 50$		$\mathrm{Tol} = 10^{-9}$	$\mu = 10^{-8}$
Optimization Results			
$\mathrm{Iterations} = 5000$		$\mathrm{Function\ Evaluations} = 250001$	
$\mathrm{J} = 38.991 \times 10^{-8}$		$\mathrm{J}_1 = 0$ $\mathrm{J}_2 = 0$	$\mathrm{J}_3 = 38.991$
$\mathrm{Perimeter} = 20.794$		$\mathrm{Area} = 28.03$	

With the utilization of ReSech activation functions in the hidden layer, our algorithm successfully approximates the convex feasible region formed by linear inequality constraints. However, the zero-valued contour of the form-free neural constitutive constraint, obtained with the utilization of the ReSech activation function, is a rather curly curve compared to one obtained with the utilization of the ReLU activation function. In other words, single-output NN with ReSech activation functions may fail to generate the form-free neural constitutive constraint forming the regular polygonal feasible region unless very large number of sample points are used together with large number of neurons in hidden layer of the NN. Thus, this example can be considered as another illustration of the importance of the ReLU activation functions for the identification of the polygonal or polyhedral convex feasible regions.

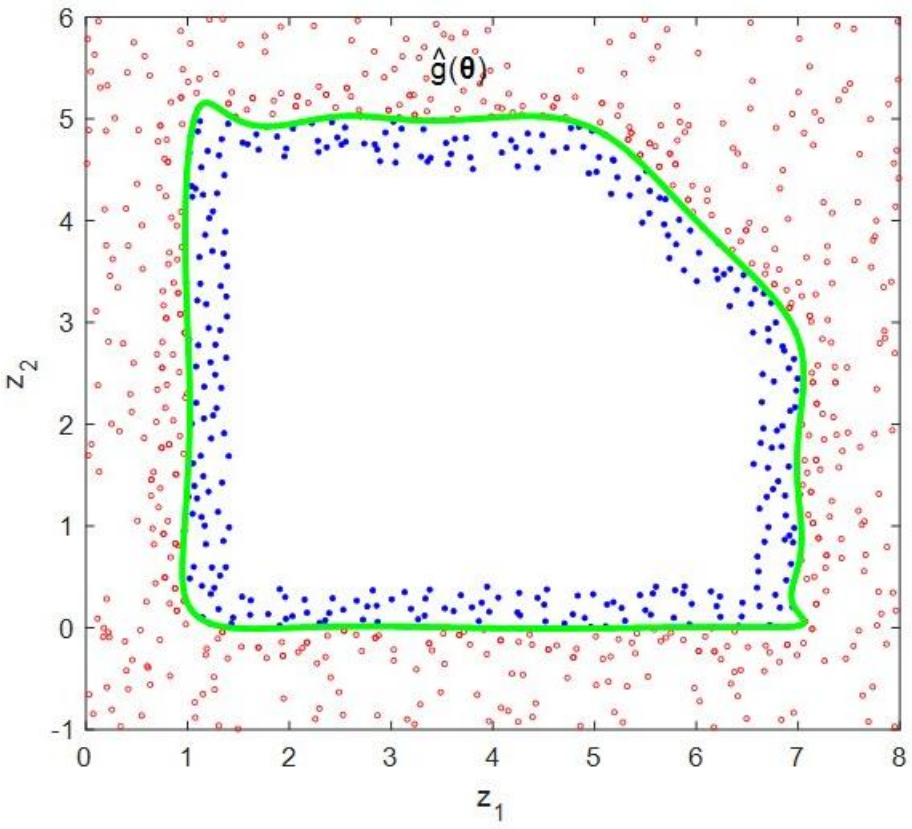

Figure 9.4. Optimal single neural constitutive constraint identified using ReSech activation function, superimposed with the boundary sets of feasible and infeasible points forming a polygonal convex feasible region.

9.3. Learning a Polygonal Convex Feasible Region via Multiple-Output NN with ReLU Activation Functions

In this section, we will demonstrate our algorithm's ability to detect a polygonal convex feasible region formed by four bound constraints and one linear inequality constraint by employing a NN with multiple outputs and equipped with ReLU activation functions.

Five inequality constraints which will be treated as unknowns and will be detected by our NN-based algorithm had been given in Equation (9.3).

NN topology used for this two-dimensional example includes ten neurons in the hidden layer and five neurons in the output layer, i.e., $\mathrm{N}^{\mathcal{H}} = 10$ and $\mathrm{N}^{\mathcal{O}} = 5$, and thus, the

number of optimization decision variables becomes $|\boldsymbol{\theta}| = 85$. We will employ ReLU and linear activation function in the hidden and output neurons, respectively. These activation functions had explicitly been given in Section 9.1 with Equation (9.4).

DE optimizer is employed to solve this example and SLHS is employed to generate the sample points as elucidated in Section 9.1. Information on the sampling, NN settings, and optimization, as well as the results are all presented in Table 9.3.

Table 9.3. Sampling, NN settings, optimization information and the results.

Sampling Information			
$\mathbf{z}^{\mathrm{LB}} = [0 \ -1]$		$\mathbf{z}^{\mathrm{UB}} = [+8 \ +6]$	
$\mathrm{N} = 2000$		$\mathrm{N}^{\mathrm{f}} = 1017$	$\mathrm{N}^{\mathrm{i}} = 1008$
$\mathbf{Z} \in \mathbb{R}^{2000\times 2}$		$\mathbf{X} \in \mathbb{R}^{1017\times 2}$	$\mathbf{Y} \in \mathbb{R}^{1008\times 2}$
		$\mathrm{p}_{\mathbf{X}} = 1$	$\mathrm{p}_{\mathbf{Y}} = 0.5$
$\mathrm{N}^{\mathrm{b}} = 649$		$\mathrm{N}^{\mathrm{fb}} = 296$	$\mathrm{N}^{\mathrm{ib}} = 353$
$\mathbf{Z} \in \mathbb{R}^{649\times 2}$		$\mathbf{X}^{\mathrm{b}} \in \mathbb{R}^{296\times 2}$	$\mathbf{Y}^{\mathrm{b}} \in \mathbb{R}^{353\times 2}$
NN and Optimization Settings			
$\mathrm{D} = 2$	$\mathrm{N}^{\mathcal{H}} = 10$	$\mathrm{N}^{\mathrm{Con}} = \mathrm{N}^{\mathcal{O}} = 5$	$\|\boldsymbol{\theta}\| = 85$
$\rho = 500$		$\mathbf{z}^{\mathrm{L}} = [-1 \ -1]$	$\mathbf{z}^{\mathrm{U}} = [+1 \ +1]$
$\mathrm{NP} = 70$		$\mathrm{Tol} = 10^{-9}$	$\mu = 10^{-5}$
Optimization Results			
$\mathrm{Iterations} = 23738$		$\mathrm{Function\ Evaluations} = 1642440$	
$\mathrm{J} = 11.765 \times 10^{-5}$		$\mathrm{J}_1 = 0$	$\mathrm{J}_2 = 0$ $\mathrm{J}_3 = 11.765$
$\mathrm{Perimeter} = 20.878$		$\mathrm{Area} = 28.142$	

Figure 9.5a presents the aggregated constraint, $\mathbb{C}(\boldsymbol{\theta})$, with the two-dimensional region formed by the boundary sets of the feasible and infeasible points. Figure 9.5b provides the identified individual neural constitutive constraints, each of which corresponds to one output neuron of the NN as explained at the onset of this chapter, together with the aggregated constraint by the blue, red, cyan, yellow, pink lines and by black dashed line without the boundary sets of feasible and infeasible points. It is possible

to observe from Figure 9.5a that the identified aggregated neural constitutive constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points included in the boundary sets. The zero values of $J_1$and J_2 in Table 9.3 imply this as well.

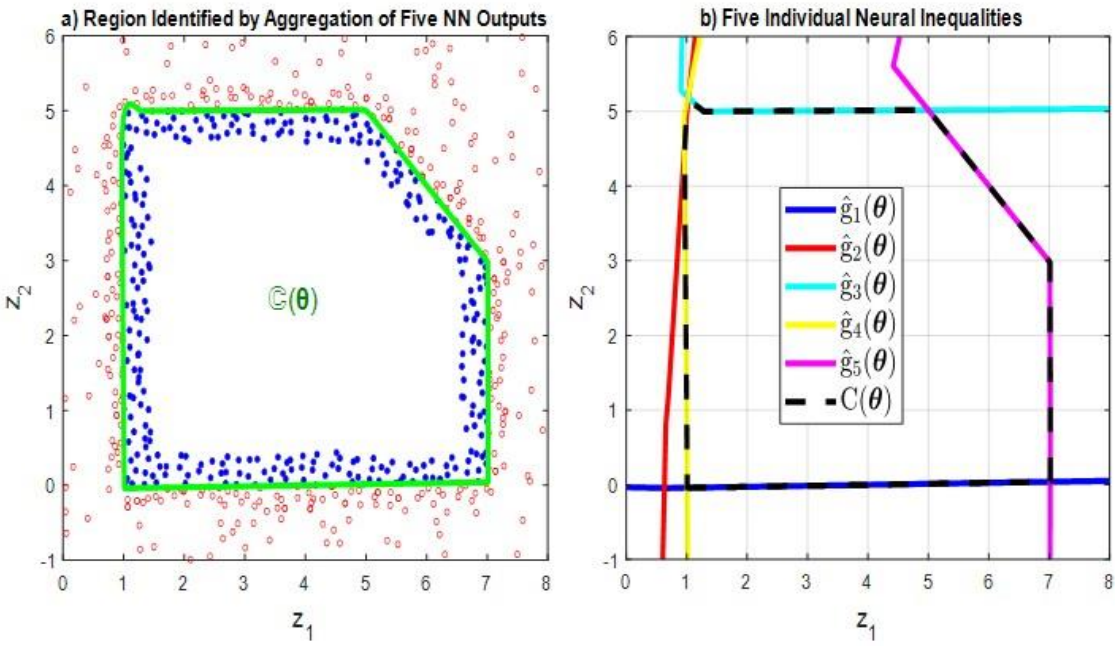

Figure 9.5. Aggregated neural constitutive constraint and individual neural constraints identified via ReLU activation function.

By the deployment of multiple-output NN with ReLU activation function, our algorithm can approximate the polygonal convex feasible region with four neural constitutive constraints, denoted by the blue, cyan, yellow, and pink lines in Figure 9.5b, even though NN topology is adjusted to generate five form-free neural constitutive constraints with five output neurons. As can be observed from Figure 9.5b that our algorithm produces the fifth neural constitutive constraint, $\hat{g}_5(\boldsymbol{\theta})$, with the capability to approximate one bound constraint and one linear inequality constraint. In other words, the zero-valued contour of $\hat{g}_5(\boldsymbol{\theta})$ is composed of two linear pieces and the second neural constitutive constraint, $\hat{g}_2(\boldsymbol{\theta})$, denoted by red line, remains a redundant constitutive constraint which does not have any effect on the constraint learning for this example. Thus, if we had used four NN outputs, we would have obtained a similar picture, possibly without a redundant constraint. We say "possibly" since in Section 9.1 we had

demonstrated that it was possible to identify the region even a single-output NN with ReLU activation function.

9.4. Learning a Polygonal Convex Feasible Region via Multiple-Output ELM with ReLU Activation Functions

In this section, we will demonstrate our algorithm's ability to detect a polygonal convex feasible region formed by four bound constraints and one linear inequality constraint by employing an ELM with multiple outputs and equipped with ReLU activation functions.

Five inequality constraints which will be treated as unknowns and will be detected by our ELM-based algorithm had been given in Equation (9.3).

ELM topology used for this two-dimensional example includes ten neurons in the hidden layer and five neurons in the output layer, i.e., $\mathrm{N}^{\mathcal{H}} = 10$ and $\mathrm{N}^{\mathcal{O}} = 5$, and output biasses are used for this case. Therefore, the number of optimization decision variables becomes $|\boldsymbol{\theta}| = 55$. We will employ ReLU and linear activation function in the hidden and output neurons, respectively. These activation functions had explicitly been given for a scalar input variable in Section 9.1 with Equation (9.4).

DE optimizer is employed to solve this example and SLHS is employed to generate the sample points as elucidated in Section 9.1. Information on the sampling, NN settings, and optimization, as well as the results are all presented in Table 9.4.

Figure 9.6a presents the aggregated constraint, $\mathbb{C}(\boldsymbol{\theta})$, with the two-dimensional region formed by the boundary sets of the feasible and infeasible points. Figure 9.6b provides the identified individual neural constitutive constraints, each of which corresponds to one output neuron of the ELM as explained at the onset of this chapter, together with the aggregated constraint by the blue, red, cyan, yellow, pink lines and by black dashed line without the boundary sets of feasible and infeasible points. It is possible to observe from Figure 9.6a that the identified aggregated neural constitutive constraint does not lead to any violations at the boundary and thus satisfies all the feasible and

infeasible points included in the boundary sets. The zero values of $J_1$and J_2 in Table 9.4 imply this as well.

Table 9.4. Sampling, ELM settings, optimization information and the results.

Sampling Information		
$\mathbf{z}^{\mathrm{LB}} = [0\ -1]$	$\mathbf{z}^{\mathrm{UB}} = [+8\ \ +6]$	
$\mathrm{N} = 2000$	$\mathrm{N}^{\mathrm{f}} = 1009$	$\mathrm{N}^{\mathrm{i}} = 1016$
$\mathbf{Z} \in \mathbb{R}^{2000\times 2}$	$\mathbf{X} \in \mathbb{R}^{1009\times 2}$	$\mathbf{Y} \in \mathbb{R}^{1016\times 2}$
	$\mathrm{p}_{\mathbf{X}} = 1$	$\mathrm{p}_{\mathbf{Y}} = 0.5$
$\mathrm{N}^{\mathrm{b}} = 690$	$\mathrm{N}^{\mathrm{fb}} = 291$	$\mathrm{N}^{\mathrm{ib}} = 399$
$\mathbf{Z} \in \mathbb{R}^{690\times 2}$	$\mathbf{X}^{\mathrm{b}} \in \mathbb{R}^{291\times 2}$	$\mathbf{Y}^{\mathrm{b}} \in \mathbb{R}^{399\times 2}$
ELM and Optimization Settings		
$\mathrm{D} = 2$ $\quad \mathrm{N}^{\mathcal{H}} = 10$	$\mathrm{N}^{\mathrm{Con}} = \mathrm{N}^{\mathcal{O}} = 5$	$\lvert\boldsymbol{\theta}\rvert = 55$
$\rho = 500$	$\mathbf{z}^{\mathrm{L}} = [-1\ -1]$	$\mathbf{z}^{\mathrm{U}} = [+1\ \ +1]$
$\mathrm{NP} = 40$	$\mathrm{Tol} = 10^{-9}$	$\mu = 10^{-5}$
Optimization Results		
$\mathrm{Iterations} = 40680$	$\mathrm{Function\ Evaluations} = 1336299$	
$\mathrm{J} = 8.294 \times 10^{-5}$	$\mathrm{J}_1 = 0$ $\quad \mathrm{J}_2 = 0$	$\mathrm{J}_3 = 8.294$
$\mathrm{Perimeter} = 20.715$	$\mathrm{Area} = 27.916$	

As can be observed from Figure 9.6b, our algorithm successfully approximates the polygonal convex feasible region mainly with the four neural constitutive constraints. The third neural constitutive constraint, $\hat{\mathrm{g}}_3(\boldsymbol{\theta})$, contributes to the detection of the polygonal convex feasible region (i.e., it is not redundant) with its small section. As in previous section that involves a NN with the identical topology for the same region, our algorithm generates one of the neural constitutive constraints, $\hat{\mathrm{g}}_1(\boldsymbol{\theta})$, as the approximator of two linear inequality constraints, i.e., a neural constitutive constraint which is composed of two major linear pieces.

By the deployment of ELM, we reduce the number of optimization decision variables compared to the NN with the same topology, while preserving the approximation

capability of NN. If we had employed a NN with identical topology as in previous section, the number of the optimization decision variables would have become $|\boldsymbol{\theta}| = 85$. In a nutshell, this section proves that it is possible to catch the approximation ability of NN even with randomly-assigned hidden-layer weights and biasses. Thus, many literature works that use NN blindfoldedly must be questioned.

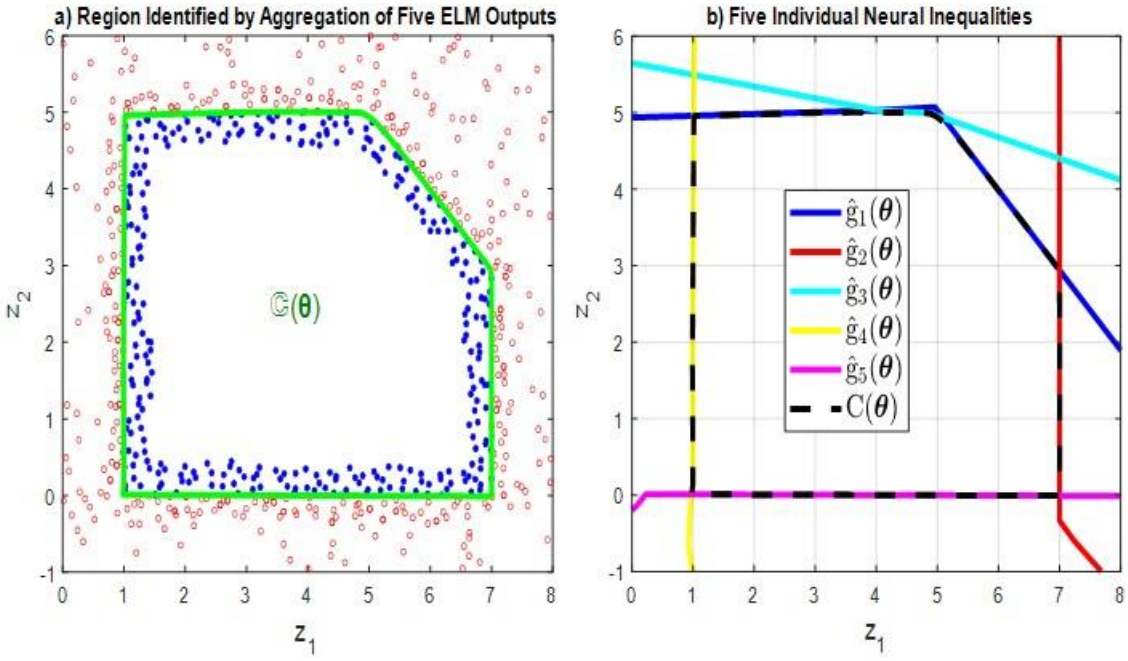

Figure 9.6. Aggregated neural constitutive constraint and individual neural constraints identified via multiple-output ELM with ReLU activation function.

9.5. Learning a Non-Convex and Disjoint Region via Single-Output NN

In this section, we will demonstrate our algorithm's ability to detect a non-convex feasible and disjoint infeasible region formed by four bounds, one linear, one circular, and one parabolic inequality constraints by employing a NN with single output and equipped with a new and unorthodox activation function.

Seven inequality constraints which will be treated as unknowns and will be detected by our NN-based algorithm are given by the following equations:

$$g_1: \quad +z_1 - 7 \leq 0, \tag{9.6a}$$

$$g_2: \quad -z_1 + 1 \leq 0, \tag{9.6b}$$

$$g_3: \quad +z_2 - 5 \leq 0, \tag{9.6c}$$

$$g_4: \quad -z_2 \leq 0, \tag{9.6d}$$

$$g_5: \quad +z_1 + z_2 - 10 \leq 0, \tag{9.6e}$$

$$g_6: \quad -(z_1 - 2.5)^2 - (z_2 - 1.5)^2 + 1 \leq 0, \tag{9.6f}$$

$$g_7: \quad -10(z_1 - 5)^2 + z_2 - 2 \leq 0. \tag{9.6g}$$

NN topology used for this two-dimensional example includes twenty-five nodes (neurons) in the hidden layer and single neuron in the output layer, i.e., $N^{\mathcal{H}} = 25$ and $N^{\mathcal{O}} = 1$, and thus, the number of optimization decision variables becomes $|\boldsymbol{\theta}| = 101$. We will employ an activation function, which is created by combining the parametric version of the radial-basis activation function with the exponential function, in the hidden neurons and linear activation function in the output neuron. The values of the parameters of the activation function used in the hidden neurons are approximate and obtained through trial-error. This activation function together with the values of its parameters is given in the following equation:

$$\text{Unorthodox Activation Function}: \quad \varphi_{\mathcal{H}}(z) = e^{z}\left(2e^{-16z^2} - 1\right). \tag{9.7}$$

This activation function is considered as unorthodox since, contrary to most of the activation functions in the literature, it is nonlinearly decreasing with increasing input, z, with a smooth transition to Gaussian peak at the middle, at $z = 0$.

We use CMA-ES optimizer to solve this example. SLHS is employed to generate the sample points as elucidated in Section 9.1. For this example, the inputs to NN are mean-centered and approximately normalized between −1 and +1. We set the weighting parameter of the third sub-objective function to zero. Information on the sampling, NN settings, and optimization, as well as the results are all presented in Table 9.5.

Figure 9.7 presents the form-free implicit constitutive constraint, $\hat{g}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour), identified with the NN, by the light green over the region formed by the boundary sets of feasible (blue) and infeasible (red) points. It is possible to observe

from Figure 9.7 that the identified neural constitutive constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points included in the boundary sets. The zero values of $J_1$and J_2 in Table 9.5 imply this as well.

Table 9.5. Sampling, NN settings, optimization information and the results.

Sampling Information		
$\mathbf{z}^{\mathrm{LB}} = [+0.75 \ -0.25]$	$\mathbf{z}^{\mathrm{UB}} = [+7.25 \ +5.25]$	
$\mathrm{N} = 1500$	$\mathrm{N}^{\mathrm{f}} = 970$	$\mathrm{N}^{\mathrm{i}} = 551$
$\mathbf{Z} \in \mathbb{R}^{1500\times 2}$	$\mathbf{X} \in \mathbb{R}^{970\times 2}$	$\mathbf{Y} \in \mathbb{R}^{551\times 2}$
	$\mathrm{p}_{\mathbf{X}} = 1$	$\mathrm{p}_{\mathbf{Y}} = 1$
$\mathrm{N}^{\mathrm{b}} = 1071$	$\mathrm{N}^{\mathrm{fb}} = 454$	$\mathrm{N}^{\mathrm{ib}} = 617$
$\mathbf{Z} \in \mathbb{R}^{1071\times 2}$	$\mathbf{X}^{\mathrm{b}} \in \mathbb{R}^{454\times 2}$	$\mathbf{Y}^{\mathrm{b}} \in \mathbb{R}^{617\times 2}$
NN and Optimization Settings		
$\mathrm{D} = 2$ $\mathrm{N}^{\mathcal{H}} = 25$	$\mathrm{N}^{\mathrm{Con}} = \mathrm{N}^{\mathcal{O}} = 1$	$\lvert\boldsymbol{\theta}\rvert = 101$
$\mathbf{z}^{\mathrm{L}} = [-5 \ -5]$	$\mathbf{z}^{\mathrm{U}} = [+5 \ +5]$	
$\mathrm{NP} = 100$	$\mathrm{Tol} = 10^{-9}$	$\mu = 0$
Optimization Results		
$\mathrm{Iterations} = 3246$	$\mathrm{Function\ Evaluations} = 324601$	
$\mathrm{J} = 0$	$\mathrm{J}_1 = 0$ $\mathrm{J}_2 = 0$	$\mathrm{J}_3 = 32.428$
$\mathrm{Perimeter} = 31.312$	$\mathrm{Area} = 22.872$	

By the deployment of NN equipped with the activation function newly introduced, our algorithm successfully approximates the non-convex feasible and disjoint infeasible region, which can be considered much more complex region compared to the polygonal feasible region presented in the previous sections, formed by seven inequality constraints. The use of other well-known activation functions, such as sigmoid, hyperbolic tangent, radial-basis, etc., is also possible. However, this unorthodox activation function provided much faster identification of this complex region. On the other hand, the number of optimization decision variables becomes $|\boldsymbol{\theta}| = 101$ with the utilization of NN with single-output. To reduce the number of optimization decision variables, we will deploy ELM with

single-output by the next section, while attempting to retain the same approximation ability, for this non-convex feasible and disjoint infeasible region.

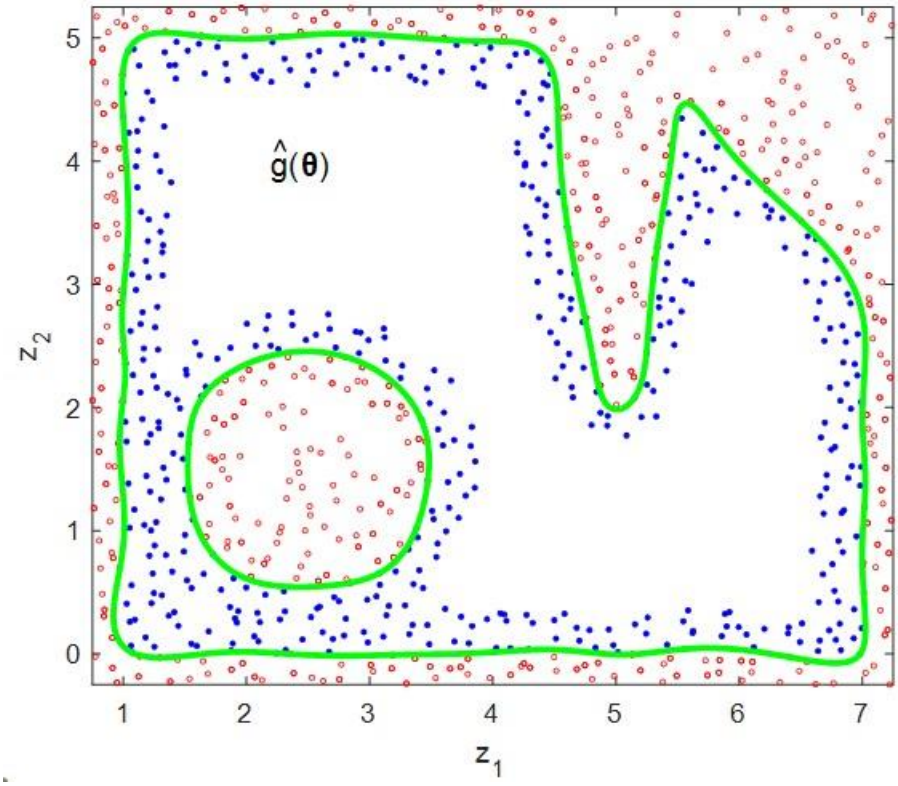

Figure 9.7. Optimal single neural constitutive constraint identified via single-output NN with unorthodox activation function, superimposed with the boundary sets of feasible and infeasible points forming a non-convex feasible and disjoint infeasible region.

9.6. Learning a Non-Convex and Disjoint Region via Single-Output ELM

In this section, we will demonstrate our algorithm's ability to detect a non-convex feasible and disjoint infeasible region, presented in the previous section, by employing an ELM with single output and equipped with the unorthodox activation function introduced in the previous section.

Seven inequality constraints which will be treated as unknowns and will be detected by our ELM-based algorithm had been given in Section 9.5 with Equation (9.7).

ELM topology used for this two-dimensional example includes fifty neurons in the hidden layer and one neuron in the output layer, i.e., $\mathrm{N}^{\mathcal{H}} = 50$ and $\mathrm{N}^{\mathcal{O}} = 1$, and output bias is not used. Therefore, the number of optimization decision variables becomes $|\boldsymbol{\theta}| = 50$. We will employ the unorthodox activation function in the hidden neurons and linear activation function in the output neuron. These activation functions for a scalar input variable had explicitly been given in Section 9.5 with Equation (9.7) and in Section 9.1 with Equation (9.4b), respectively.

We use CMA-ES optimizer without imposing the lower and upper bounds on the optimization decision variables. SLHS is employed to generate the sample points as elucidated in Section 9.1. The inputs to ELM are normalized as in the previous section and the weighting parameter of the third sub-objective function is set to zero. Information on the sampling, ELM settings, and optimization, as well as the results are all presented in Table 9.6.

Figure 9.8 presents the form-free implicit constitutive constraint, $\hat{\mathrm{g}}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour), identified with the ELM, by the light green over the region formed by the boundary sets of feasible (blue) and infeasible (red) points. It is possible to observe from Figure 9.8 that the identified neural constitutive constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points included in the boundary sets. The zero values of $\mathrm{J}_1$and J_2 in Table 9.6 imply this as well.

By the deployment of ELM, equipped with the unorthodox activation function, our algorithm successfully approximates the non-convex feasible and disjoint infeasible region formed by seven inequality constraints. Although the number of the hidden neurons is doubled compared to the previous section, the use of ELM considerably reduces the number of optimization-decision variables, from 101 to 50. Additionally, this section proves that the utilization of ELM together with the unorthodox activation function is not only suitable for the detection of such complex regions but also allows our algorithm to execute much faster identification compared to NN case of the previous section. Therefore, if the aim is to approximate such regions, it can be better to first consider the utilization of ELM rather than the deployment of NN.

Table 9.6. Sampling, ELM settings, optimization information and the results.

Sampling Information		
$\mathbf{z}^{\mathrm{LB}} = [+0.75 \;\; -0.25]$	$\mathbf{z}^{\mathrm{UB}} = [+7.25 \;\; +5.25]$	
$\mathrm{N} = 1500$	$\mathrm{N}^{\mathrm{f}} = 971$	$\mathrm{N}^{\mathrm{i}} = 550$
$\mathbf{Z} \in \mathbb{R}^{1500\times 2}$	$\mathbf{X} \in \mathbb{R}^{971\times 2}$	$\mathbf{Y} \in \mathbb{R}^{550\times 2}$
	$\mathrm{p}_{\mathbf{X}} = 1$	$\mathrm{p}_{\mathbf{Y}} = 1$
$\mathrm{N}^{\mathrm{b}} = 1067$	$\mathrm{N}^{\mathrm{fb}} = 453$	$\mathrm{N}^{\mathrm{ib}} = 614$
$\mathbf{Z} \in \mathbb{R}^{1067\times 2}$	$\mathbf{X}^{\mathrm{b}} \in \mathbb{R}^{453\times 2}$	$\mathbf{Y}^{\mathrm{b}} \in \mathbb{R}^{614\times 2}$
ELM and Optimization Settings		
$\mathrm{D} = 2$ $\quad \mathrm{N}^{\mathcal{H}} = 50$	$\mathrm{N}^{\mathrm{Con}} = \mathrm{N}^{\mathcal{O}} = 1$	$\lvert\boldsymbol{\theta}\rvert = 50$
$\mathrm{NP} = 100$	$\mathrm{Tol} = 10^{-9}$	$\mu = 0$
Optimization Results		
$\mathrm{Iterations} = 2767$	$\mathrm{Function\ Evaluations} = 276701$	
$\mathrm{J} = 0$	$\mathrm{J}_1 = 0$ $\quad \mathrm{J}_2 = 0$	$\mathrm{J}_3 = 22.109$
$\mathrm{Perimeter} = 31.189$	$\mathrm{Area} = 22.733$	

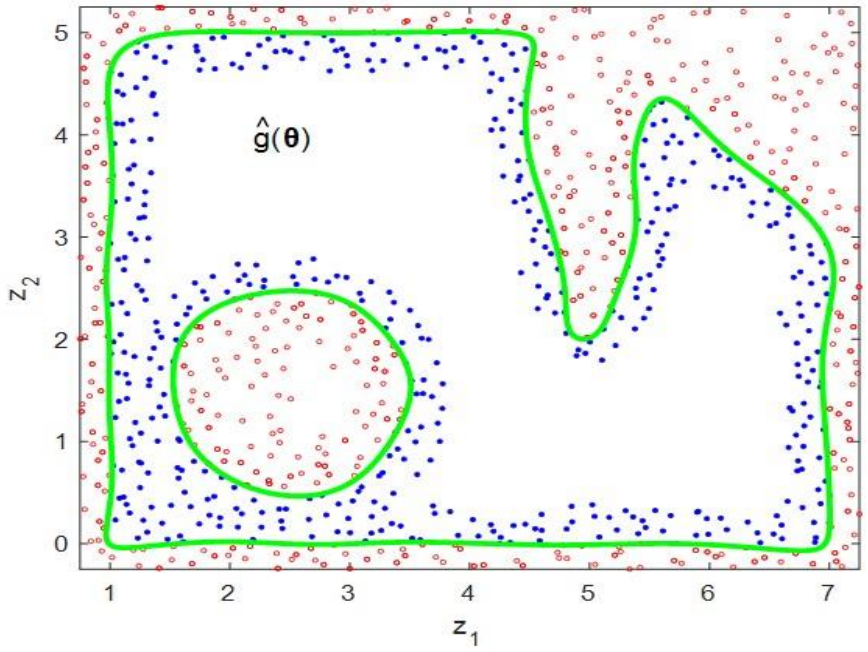

Figure 9.8. Optimal single neural constitutive constraint identified via single-output ELM with unorthodox activation function, superimposed with the boundary sets of feasible and infeasible points forming a non-convex feasible and disjoint infeasible region.

9.7. Learning a Non-Convex and Disjoint Region via Multiple-Output NN

In this section, we will demonstrate our algorithm's ability to detect a non-convex feasible and disjoint infeasible region presented in the previous section by employing a NN with multiple output and equipped with the unorthodox activation function introduced in the previous section.

Seven inequality constraints which will be treated as unknowns and will be detected by our NN-based algorithm had been given in Section 9.5 with Equation (9.7).

NN topology used for this two-dimensional example includes fifteen neurons in the hidden layer and three neurons in the output layer, i.e., $\mathrm{N}^{\mathcal{H}} = 15$ and $\mathrm{N}^{\mathcal{O}} = 3$, and thus, the number of optimization decision variables becomes $|\boldsymbol{\theta}| = 93$. We will employ the unorthodox activation function in the hidden neurons and linear activation function in the output neurons. These activation functions for a scalar input variable had explicitly been given in Section 9.5 with Equation (9.7) and in Section 9.1 with Equation (9.4b), respectively.

We use CMA-ES optimizer without imposing the lower and upper bounds on the optimization-decision variables. SLHS is employed to generate the sample points as elucidated in Section 9.1. The inputs to NN are normalized and the weighting parameter of the third sub-objective function is set to zero as in the previous section. Information on the sampling, NN settings, and optimization, as well as the results are all presented in Table 9.7.

Figure 9.9a presents the aggregated constraint, $\mathbb{C}(\boldsymbol{\theta})$, with the two-dimensional region formed by the boundary sets of the feasible and infeasible points. Figure 9.9b provides the identified individual neural constitutive constraints, each of which corresponds to one output neuron of the NN as explained at the onset of this chapter, together with the aggregated constraint by the blue, red, cyan lines and by black dashed line without the boundary sets of feasible and infeasible points.

Table 9.7. Sampling, NN settings, optimization information and the results.

Sampling Information			
$\mathbf{z}^{\mathrm{LB}} = [+0.75 \ -0.25]$		$\mathbf{z}^{\mathrm{UB}} = [+7.25 \ +5.25]$	
$\mathrm{N} = 1500$		$\mathrm{N}^{\mathrm{f}} = 975$	$\mathrm{N}^{\mathrm{i}} = 546$
$\mathbf{Z} \in \mathbb{R}^{1500\times 2}$		$\mathbf{X} \in \mathbb{R}^{975\times 2}$	$\mathbf{Y} \in \mathbb{R}^{546\times 2}$
		$\mathrm{p}_{\mathbf{X}} = 1$	$\mathrm{p}_{\mathbf{Y}} = 1$
$\mathrm{N}^{\mathrm{b}} = 1056$		$\mathrm{N}^{\mathrm{fb}} = 446$	$\mathrm{N}^{\mathrm{ib}} = 610$
$\mathbf{Z} \in \mathbb{R}^{1056\times 2}$		$\mathbf{X}^{\mathrm{b}} \in \mathbb{R}^{446\times 2}$	$\mathbf{Y}^{\mathrm{b}} \in \mathbb{R}^{610\times 2}$
NN and Optimization Settings			
$\mathrm{D} = 2$	$\mathrm{N}^{\mathcal{H}} = 15$	$\mathrm{N}^{\mathrm{Con}} = \mathrm{N}^{\mathcal{O}} = 3$	$\lvert\boldsymbol{\theta}\rvert = 93$
		$\rho = 500$	
$\mathrm{NP} = 100$		$\mathrm{Tol} = 10^{-9}$	$\mu = 0$
Optimization Results			
$\mathrm{Iterations} = 2376$		$\mathrm{Function\ Evaluations} = 237601$	
$\mathrm{J} = 0$		$\mathrm{J}_1 = 0$ $\mathrm{J}_2 = 0$	$\mathrm{J}_3 = 21.821$
$\mathrm{Perimeter} = 31.745$		$\mathrm{Area} = 22.775$	

It is possible to observe from Figure 9.9a that the identified aggregated neural constitutive constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points included in the boundary sets. The zero values of $\mathrm{J}_1$and J_2 in Table 9.7 imply this as well.

By the deployment of multiple-output NN, our algorithm successfully approximates the non-convex feasible and disjoint infeasible region with three neural constitutive constraints. As can be observed from Figure 9.9b that our algorithm produces the first neural constitutive constraint, $\hat{\mathrm{g}}_1(\boldsymbol{\theta})$, with the capability to approximate two bound constraints and one linear inequality constraint and one circular inequality constraint. Algorithm also generates the second neural constitutive constraint, $\hat{\mathrm{g}}_2(\boldsymbol{\theta})$, as the approximator of one bound constraint and one parabolic inequality constraint. To reveal the last unknown bound constraint, our algorithm uses the third neural constitutive constraint, $\hat{\mathrm{g}}_3(\boldsymbol{\theta})$, alone.

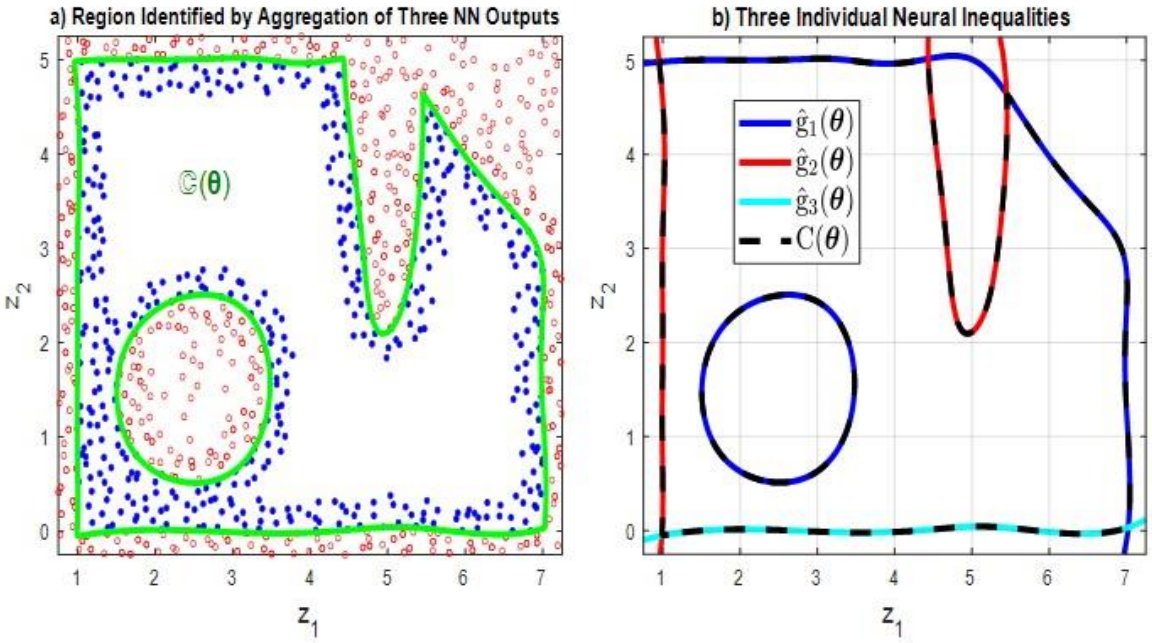

Figure 9.9. Aggregated neural constitutive constraint and individual neural constraints identified via multiple-output NN with unorthodox activation function.

10. PERIPHERY IDENTIFICATION BY CLASSIFICATION OF FEASIBLE AND INFEASIBLE REGIONS VIA MACHINE LEARNING

In this chapter, we will illustrate how to employ the classification approach for the identification of the peripheries of the feasible and infeasible regions. For this purpose, we will apply to the constraint-identification task five Machine Learning (ML) algorithms which are capable of handling classification problems, these are the Probabilistic Neural Network, k-Nearest Neighbours, Support Vector Machine, Gaussian Process Regression and Classification and Regression Trees.

As can be remembered, till now, there had been no discrete explicit labels (categorization) for the sample points as feasible or infeasible. Additionally, this thesis work had been formulated as an optimization problem (Chapter 5) and several methods (Chapter 5, Chapter 7, Chapter 9) had been embedded into the optimization problem for the identification (functional approximation) of the inequality constraints. Thus, this chapter will begin with the short explanation of how we can reconsider the constraint-identification task as a classification problem. This explanation will be valid for all the classification algorithms used in respective sections of this chapter. After explanation of the classification problem of this chapter, we will continue with the information of the non-convex feasible and disjoint infeasible regions which will be used for each classification algorithm identically. Lastly, this chapter will be completed with five sections that cover the short explanation of the five classification algorithms together with one or two examples of periphery identification by tuning the hyperparameters of these algorithms.

The classification algorithms aim at predicting the true label (group) of unseen samples by learning specific or certain rules from training samples. For the classification problems of this chapter, there exist two classes of sample points as “feasible” and “infeasible”. Thus, we can label feasible points as +1 and infeasible points as −1 to indicate their groups. For instance, our training set is the complete set of N sample points generated by one of the sampling methods elucidated in Chapter 3. We can obtain a label set for the

complete set of all points, $t^k \in \{-1, +1\}$ where $k = \{1, \dots, N\}$, and pair these labels up with all points included in the complete set. As a result, we can achieve a paired set of all points denoted by $\mathbf{Z}^{\text{Paired}} = \{\mathbf{z}^k, t^k\}$ where $k = \{1, \dots, N\}$. After training any of the classification algorithms employed in this chapter, we will check the classification algorithms with the test set which is the vectorized mesh-grid points obtained based on $\mathbf{z}^{\text{LB}}$ and $\mathbf{z}^{\text{UB}}$ of the coordinates with points much more than N, i.e., 250000 points. We will also provide the peripheries detected by using the test samples. Lastly, in some situations, where Gaussian Process Regression, for instance, is employed as classifier, the classification algorithms may not give pure integer outputs but as the fractional numbers too close to the integer labels. In such situations, we use the hyperbolic-tangent function to round such fractional numbers towards the values of −1 or +1.

Seven inequality constraints which will be treated as unknowns and will be detected only as the form of the periphery by the classification algorithms had already been given in Chapter 9 (Section 9.5 with Equation (9.7)).

MS is employed to generate the sample points as elucidated in Chapter 3. For this section, the procedure of the boundary-zone formation is not employed. Information on the sampling is presented in Table 10.1.

Table 10.1. Sampling information.

$\mathbf{z}^{\text{LB}} = [+0 \ -1]$	$\mathbf{z}^{\text{UB}} = [+8 \ +6]$	
$N = 1500$	$N^f = 600$	$N^i = 921$
$\mathbf{Z} \in \mathbb{R}^{1500\times 2}$	$\mathbf{X} \in \mathbb{R}^{600\times 2}$	$\mathbf{Y} \in \mathbb{R}^{921\times 2}$

10.1. Learning Region Boundaries via Probabilistic Neural Network

In this section, we will use the Probabilistic Neural Network (PNN) to approximate the peripheries of non-convex feasible and disjoint infeasible regions mentioned at the onset of this chapter. We will illustrate the effect of the hyperparameter of the PNN on the identification task with two different cases.

PNN is a type of the feed-forward neural network utilized for classification and pattern-recognition problems. It depends on the approximation of the parent probability distribution of classes. We will employ PNN via MATLAB's "newpnn" function. This function forms PNN as two-layer network except for the input and output layers. The first one is called as radial-basis layer and the second one competitive layer. In radial basis layer, the distance vector between each test sample and the weight matrix of radial basis layer, which includes all training samples, is estimated. Distance vector for each test sample is multiplied by the value of bias which depends on the user-defined hyperparameter "spread". The resulting vector goes through radial-basis activation function. In competitive layer, the output vector of radial basis layer is multiplied by the weight matrix of competitive layer which denotes the actual label of all training samples (Wasserman, 1993). Therefore, the hyperparameter of PNN is the value of "spread" and we will show how this hyperparameter has an effect on the detection of a complex region via two examples.

Figure 10.1 presents the non-parametric periphery (i.e., actually its zero-valued contour), identified via PNN classification with the spread value of 0.5, by the light green over the region formed by the complete sets of feasible (blue) and infeasible (red) points.

It is possible to observe from Figure 10.1 that the identified non-parametric periphery obtained with this value of the spread causes some violations at the boundary and thus does not satisfy some feasible and infeasible points. In other words, PNN classification fails to successfully classify some feasible and infeasible points (20 out of 1500 points) with this setting of spread.

Figure 10.2 presents the non-parametric periphery (i.e., actually its zero-valued contour), identified via PNN classification with the spread of 0.1, by the light green over the region formed by the complete sets of feasible (blue) and infeasible (red) points.

It is possible to observe from Figure 10.2 that the identified non-parametric periphery obtained with this spread setting does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. In other words, PNN classification successfully categorizes all feasible and infeasible points into their true groups and

provides the non-parametric periphery which can be utilized instead of the unknown inequality constraints.

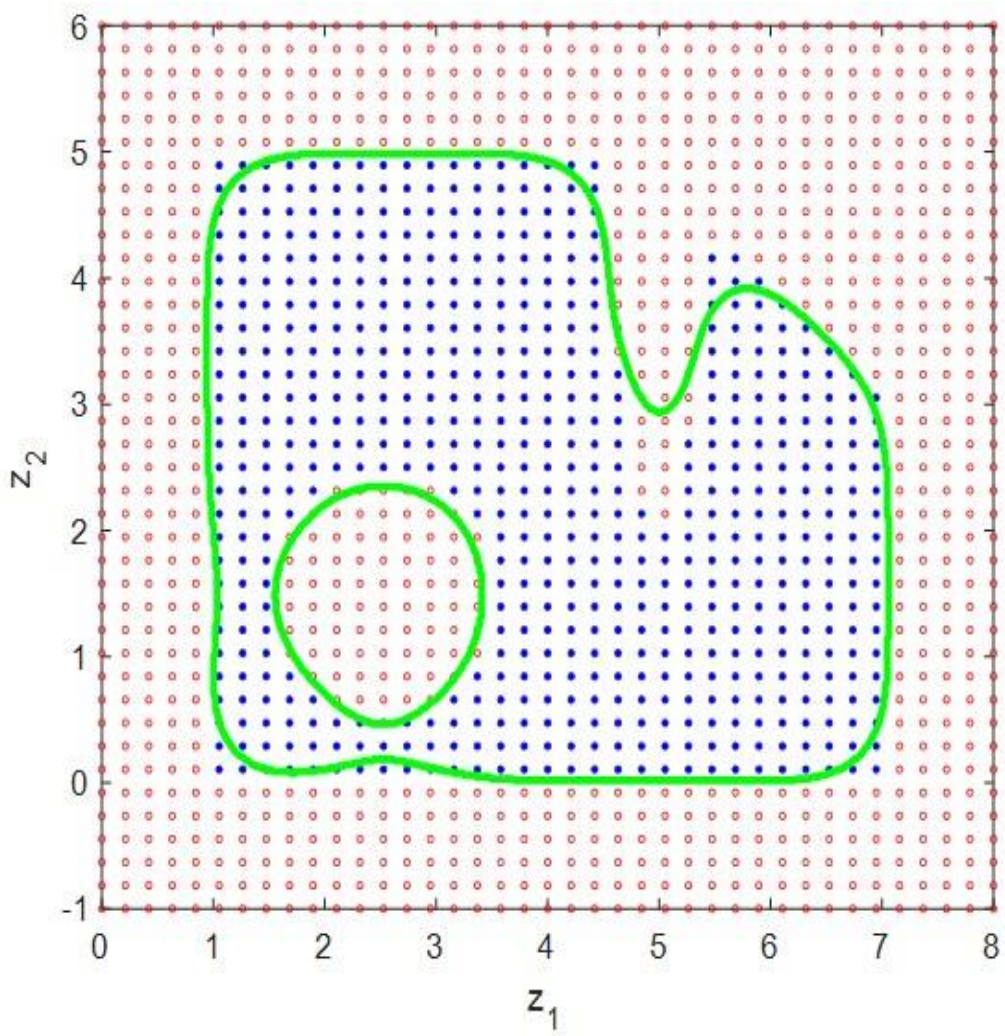

Figure 10.1. Non-parametric periphery identified via PNN classification with a spread of 0.5.

As can be observed from the presented cases (Figure 10.1 and Figure 10.2), although PNN classification becomes successful at detecting feasible and infeasible region boundary with a lower value of spread (Figure 10.2), it yields more smooth curve with higher spread value (Figure 10.1) at the expense of becoming unsuccessful. Therefore, it can be concluded that there is a trade-off between smoothness and accuracy of the identified periphery of the regions, which can be resolved by the proper selection of the PNN's "spread" hyperparameter.

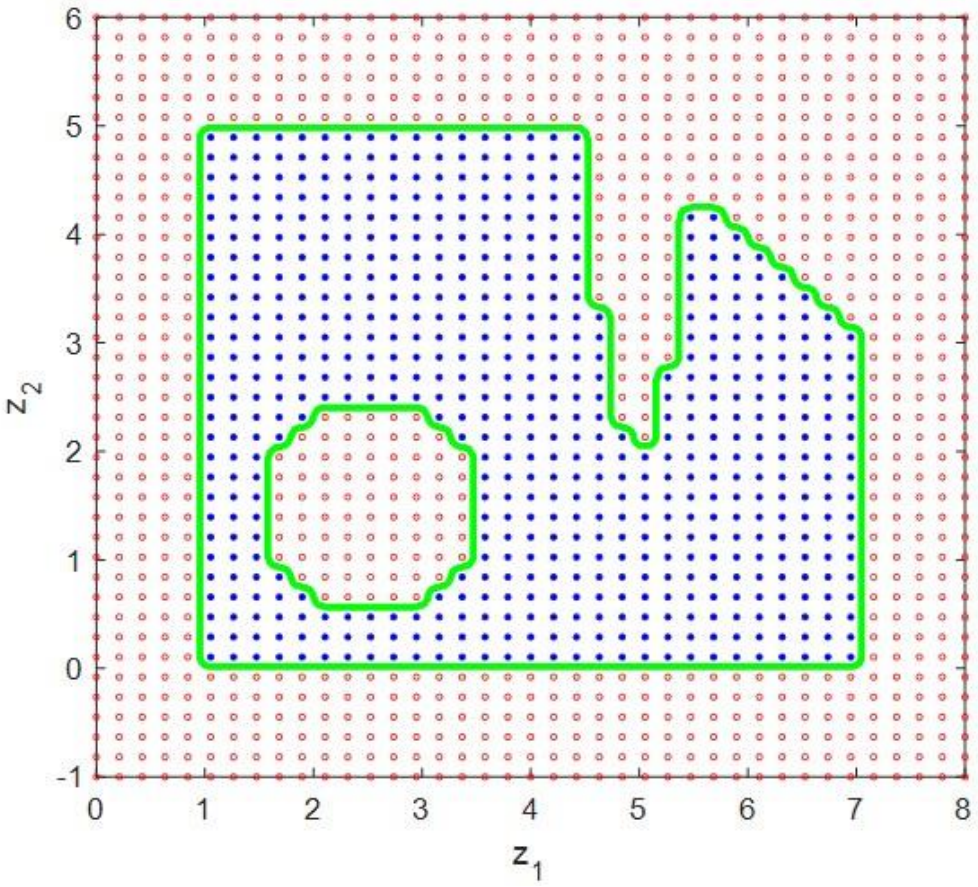

Figure 10.2. Non-parametric periphery identified via PNN classification with a spread of 0.1.

10.2. Learning Region Boundaries via k-Nearest Neighbours Classification Algorithm

In this section, we will use the k-Nearest Neighbours (k-NN) classification algorithm to approximate the peripheries of non-convex feasible and disjoint infeasible regions mentioned at the onset of this chapter. We will illustrate the effect of the hyperparameter of the k-NN on the identification task with two different cases.

k-NN classification algorithm is one of the supervised learning algorithms which uses a distance metric in order to estimate the distances between each test sample and the selected number of the closest training samples (distance of k closest training samples to each test sample) and categorize each test sample by taking the majority label of the k training samples into account (Cover and Hart, 1967). We will use the k-NN algorithm via MATLAB's "fitcknn" function which accommodates Euclidean-distance metric as default.

As can be understood from its definition, “k” is the hyperparameter of this algorithm. We will also provide the effect of the value of the hyperparameter “k” on the approximation of peripheries of a complex region via two examples.

Figure 10.3 presents the non-parametric periphery (i.e., actually its zero-valued contour), identified via k-NN classification with k = 17, by the light green over the region formed by the complete sets of feasible (blue) and infeasible (red) points. It is possible to observe from Figure 10.3 that the identified non-parametric periphery obtained with this value of “k” causes some violations at the boundary and thus does not satisfy some feasible and infeasible points. In other words, k-NN classification fails to successfully classify some feasible and infeasible points (14 out of 1500 points) with this setting of ”k”.

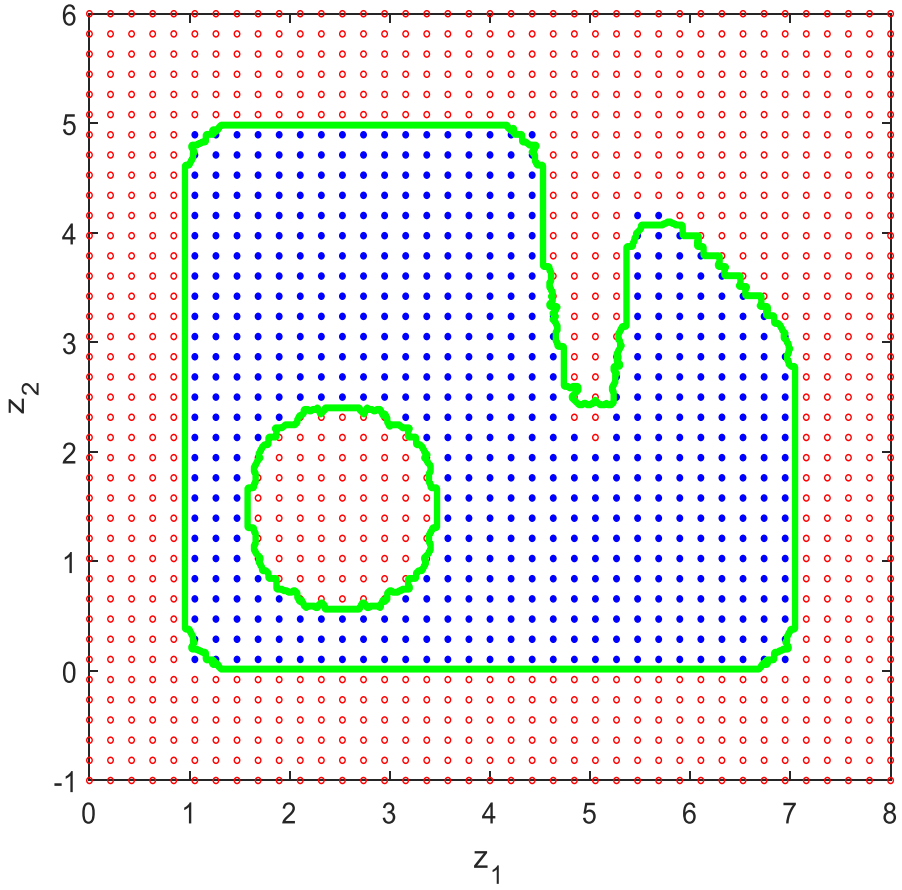

Figure 10.3. Non-parametric periphery identified via k-NN classification with k = 17.

Figure 10.4 presents the non-parametric periphery (i.e., actually its zero-valued contour), identified via k-NN classification with k = 3, by the light green over the region

formed by the complete sets of feasible (blue) and infeasible (red) points. It is possible to observe from Figure 10.4 that the identified non-parametric periphery obtained with this "k" setting does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. In other words, k-NN classification successfully categorizes all feasible and infeasible points into their true groups and provides the non-parametric periphery which can be utilized instead of the unknown inequality constraints.

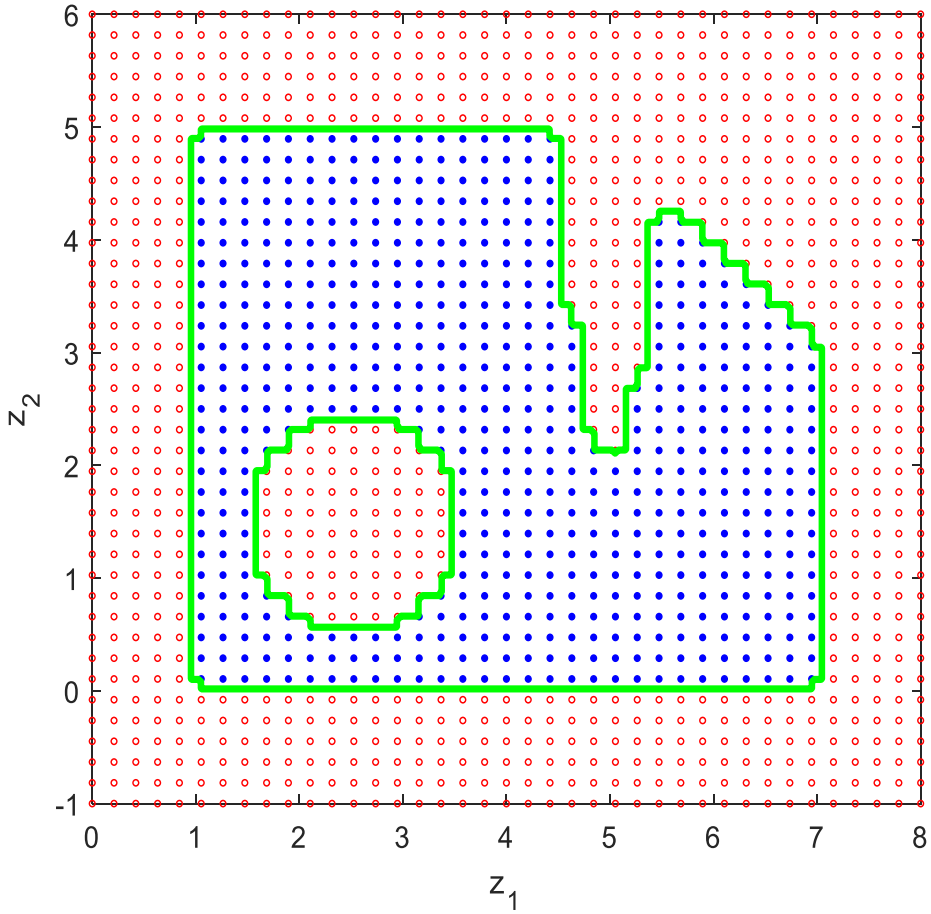

Figure 10.4. Non-parametric periphery identified via k-NN classification with k = 3.

As can be observed from the above-presented cases (Figure 10.3 and Figure 10.4), the increase in the hyperparameter "k" results in the erroneous identification of the region. Therefore, it is better to keep the value of "k" as lower as possible if the k-NN algorithm is utilized for the constraint-identification task. However, very low values of "k", such as one, may yield highly zigzagged peripheries for some complex regions.

10.3. Learning Region Boundaries via Support Vector Machine Classification

In this section, we will use the Support Vector Machine (SVM) as classifier to approximate the peripheries of non-convex feasible and disjoint infeasible regions mentioned at the onset of this chapter. We will illustrate the effect of the hyperparameter of the SVM on the identification task with two different cases.

SVM relies on discovering a hyperplane that best divides two classes included in a sample set. This separation occurs via the maximization of the margin between the support vector and two sample points, which are the closest to each other, from two groups. This maximization problem is a quadratic programming (QP) problem and QPs are solved to global optimality very fast. One of the important properties of SVM is that it uses kernel trick for samples which are not linearly separable (Ben-Hur and Weston, 2010). We will use the SVM algorithm via MATLAB's "fitcsvm" function that accommodates radial-basis function as kernel. The parameter of kernel function, called the kernel-scale factor, becomes the hyperparameter of the SVM algorithm tuned for this section.

Figure 10.5 presents the non-parametric periphery (i.e., actually its zero-valued contour), identified via SVM classification with the kernel-scale factor of one, by the light green over the region formed by the complete sets of feasible (blue) and infeasible (red) points. It is possible to observe from Figure 10.5 that the identified non-parametric periphery obtained with this value of kernel-scale factor causes some violations at the boundary and thus does not satisfy some feasible and infeasible points. In other words, SVM classification fails to successfully classify some feasible and infeasible points (27 out of 1500 points) with this setting of the kernel-scale factor.

Figure 10.6 presents the non-parametric periphery (i.e., actually its zero-valued contour), identified via SVM classification with the kernel-scale factor of 0.2, by the light green over the region formed by the complete sets of feasible (blue) and infeasible (red) points.

It is possible to observe from Figure 10.6 that the identified non-parametric periphery obtained with this value of the kernel-scale factor does not lead to any violations at the

boundary and thus satisfies all the feasible and infeasible points. In other words, SVM classification successfully categorizes all feasible and infeasible points into their true groups and provides the non-parametric periphery which can be employed instead of the unknown inequality constraints.

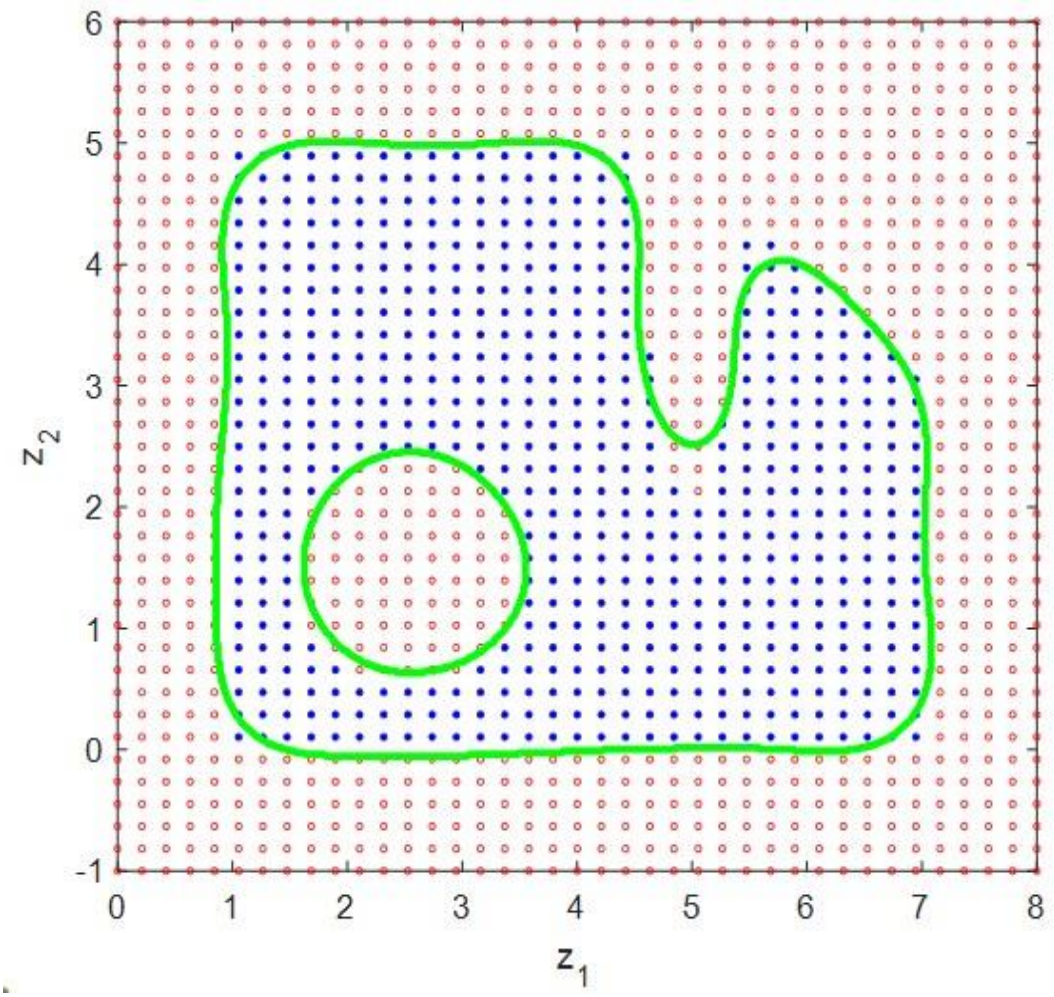

Figure 10.5. Non-parametric periphery identified via SVM classification with the kernel-scale factor of one.

As can be observed from the presented cases (Figure 10.5 and Figure 10.6), the kernel-scale factor has the effect not only on the smoothness but also on the true identification of the peripheries of this complex region. If the kernel scale is kept at around 0.2, SVM algorithm gives us the true peripheries, but with a less smooth curve. If the kernel scale is set to around unity, we can obtain a more smooth boundary at the expense of the erroneous identification of the peripheries of the complex region.

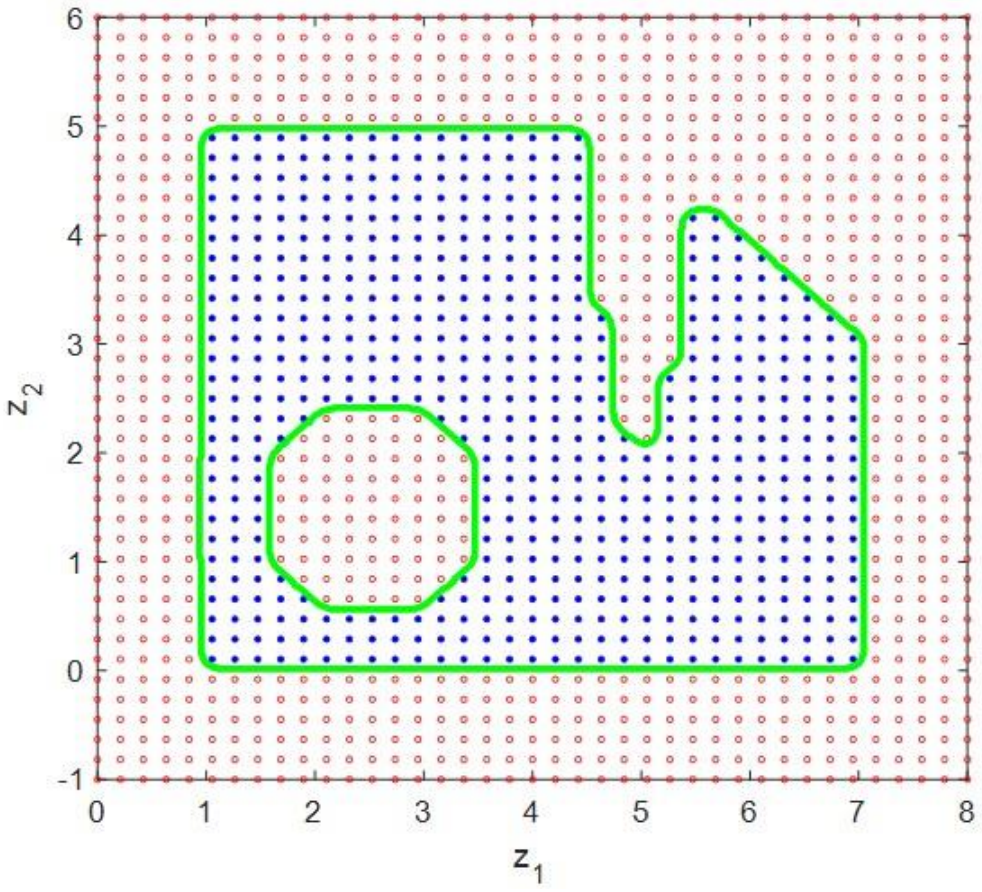

Figure 10.6. Non-parametric periphery identified via SVM classification with the kernel-scale factor of 0.2.

10.4. Learning Region Boundaries via Gaussian Process Regression

In this section, we will use the Gaussian Process Regression (GPR) as classifier to approximate the peripheries of non-convex feasible and disjoint infeasible regions mentioned at the onset of this chapter.

GPR is one of the non-parametric ML algorithms that takes advantage of a basis function and the latent samples from a gaussian process. With a gaussian process, it is assumed that any certain number of randomly selected samples from a set has the multivariate gaussian distribution expressed by the mean and covariance-kernel functions (Rasmussen and Williams, 2006). We will use the GPR algorithm via MATLAB's "fitrgp" function that employs "linear" and "matern 3/2" functions as basis and covariance-kernel functions, respectively.

Figure 10.7 presents the non-parametric periphery (i.e., actually its zero-valued contour), identified via GPR algorithm, by the light green over the region formed by the complete sets of feasible (blue) and infeasible (red) points. It is possible to observe from Figure 10.7 that the identified non-parametric periphery obtained with this basis and covariance-kernel functions does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. In other words, GPR algorithm successfully categorizes all feasible and infeasible points into their true groups and provides the non-parametric periphery which can be employed instead of the unknown inequality constraints.

As can be seen in Figure 10.7, by the utilization of GPR with its above-mentioned settings, we achieve the peripheries almost equal to the ones obtained via the SVM with the kernel scale of 0.2 (Figure 10.6). Thus, it can be concluded that they arguably show the similar approximation ability for the complex problems of the constraint identification if their settings are properly adjusted.

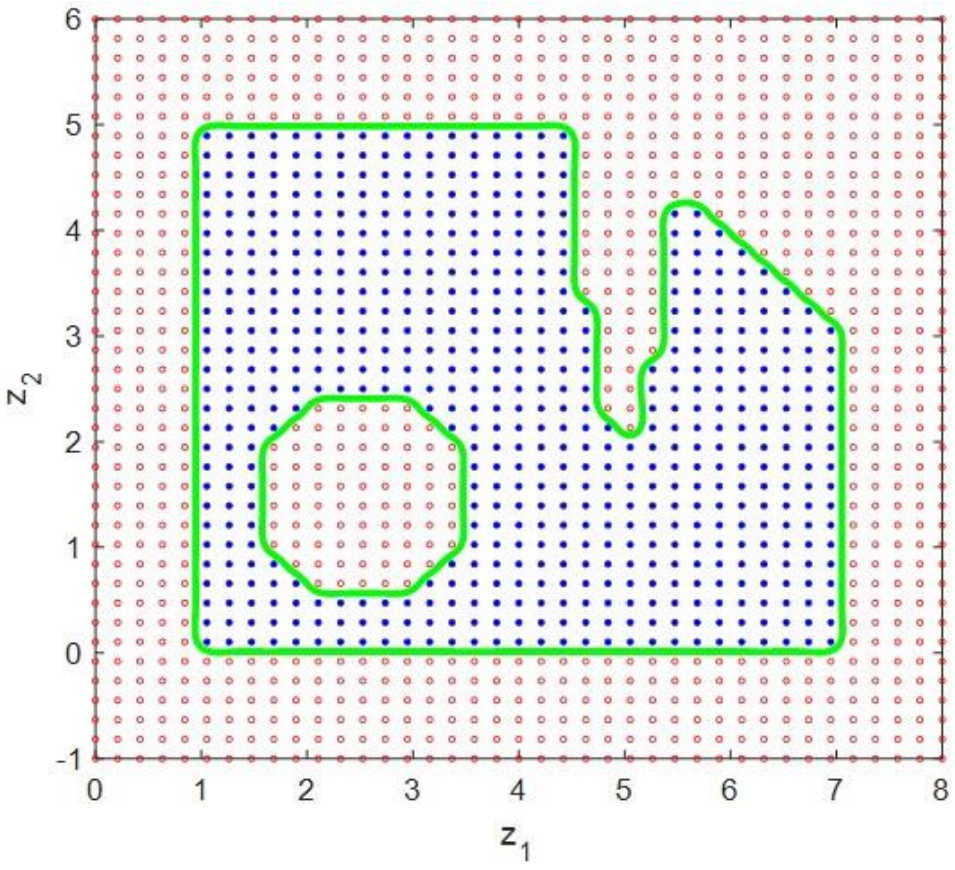

Figure 10.7. Non-parametric periphery identified via GPR algorithm.

10.5. Learning Region Boundaries via Classification and Regression Trees Algorithm

In the last section of this chapter, we will use the Classification and Regression Trees (CART) algorithm as classifier to approximate the peripheries of non-convex feasible and disjoint infeasible regions mentioned at the onset of this chapter. We will exemplify the effect of the hyperparameter of the CART on the identification task with two different cases.

CART algorithm is one of the supervised learning algorithms that can be utilized both for classification and regression problems. CART begins with a root node, which includes all training samples, and split root node and child nodes until minimum impurity is achieved in the leaf nodes. Minimum impurity attained for a leaf node means that it has only the members of one class (homogenous distribution of a class) and impurity can be measured by several metrics such as Gini Index, Entropy and Misclassification Error. Additionally, Decision Trees (DT) algorithm partitions the feature space by employing binary split or multi-way split (Breiman et al., 1984). We will use the CART algorithm via MATLAB's "fitctree" function. By this function, CART splits parent nodes and creates child nodes by using the binary-split approach. Impurity in each node is also estimated by Gini Index. There exist three hyperparameters of this algorithm such as the minimum number of samples in the parent (branch) and leaf nodes, maximum number of splits which can be included in the structure of the CART. The minimum number of samples in the leaf nodes and maximum number of splits are set to one and $N-1$, respectively. Only the minimum number of samples in the parent nodes is tuned and illustrated how to effect on the identification task for this section.

Figure 10.8 presents the non-parametric periphery (i.e., actually its zero-valued contour), identified via CART algorithm with the minimum number of samples of 16 for the parent nodes, by the light green over the region formed by the complete sets of feasible (blue) and infeasible (red) points. It is possible to observe from Figure 10.8 that the identified non-parametric periphery obtained with this value of the minimum number of samples for the parent nodes causes some violations at the boundary and thus does not satisfy some feasible and infeasible points. In other words, CART algorithm fails to

successfully classify some feasible and infeasible points (18 out of 1500 points) with this setting.

Figure 10.9 presents the non-parametric periphery (i.e., actually its zero-valued contour), identified via CART algorithm with the minimum number of samples of one for the parent nodes, by the light green over the region formed by the complete sets of feasible (blue) and infeasible (red) points. It is possible to observe from Figure 10.9 that the identified non-parametric periphery obtained with this value of the minimum number of samples for parent nodes does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. In other words, CART algorithm successfully categorizes all feasible and infeasible points into their true groups and provides the non-parametric periphery which can be employed instead of the unknown inequality constraints.

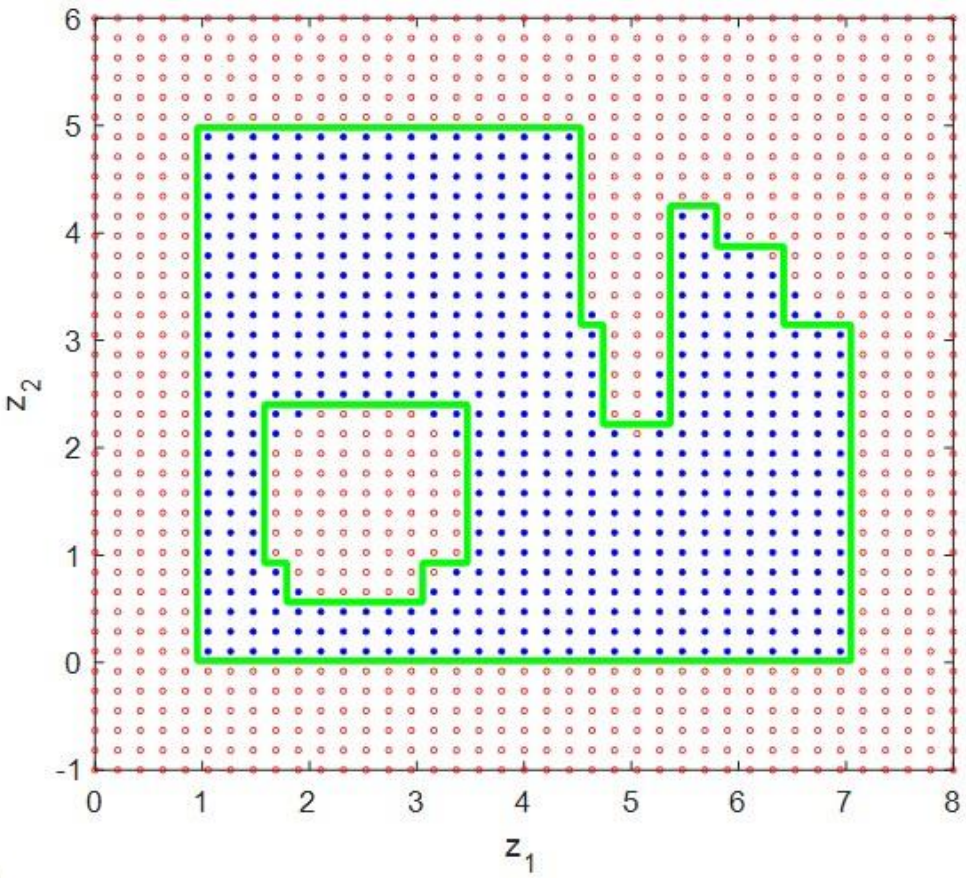

Figure 10.8. Non-parametric periphery identified via CART algorithm with minimum number of samples of 16 for parent nodes.

As can be observed from the first case (Figure 10.8), the high minimum-number of samples in the parent nodes gives the erroneous identification of this region. On the other hand, removing the restriction in terms of the maximum minimality of the parent size allows the CART algorithm to successfully approximate this complex region (Figure 10.9). Arguably, had we set the minimum number of samples for parent nodes to greater than 16, the CART algorithm would have given erroneous identification that misclassifies more samples compared to the first case.

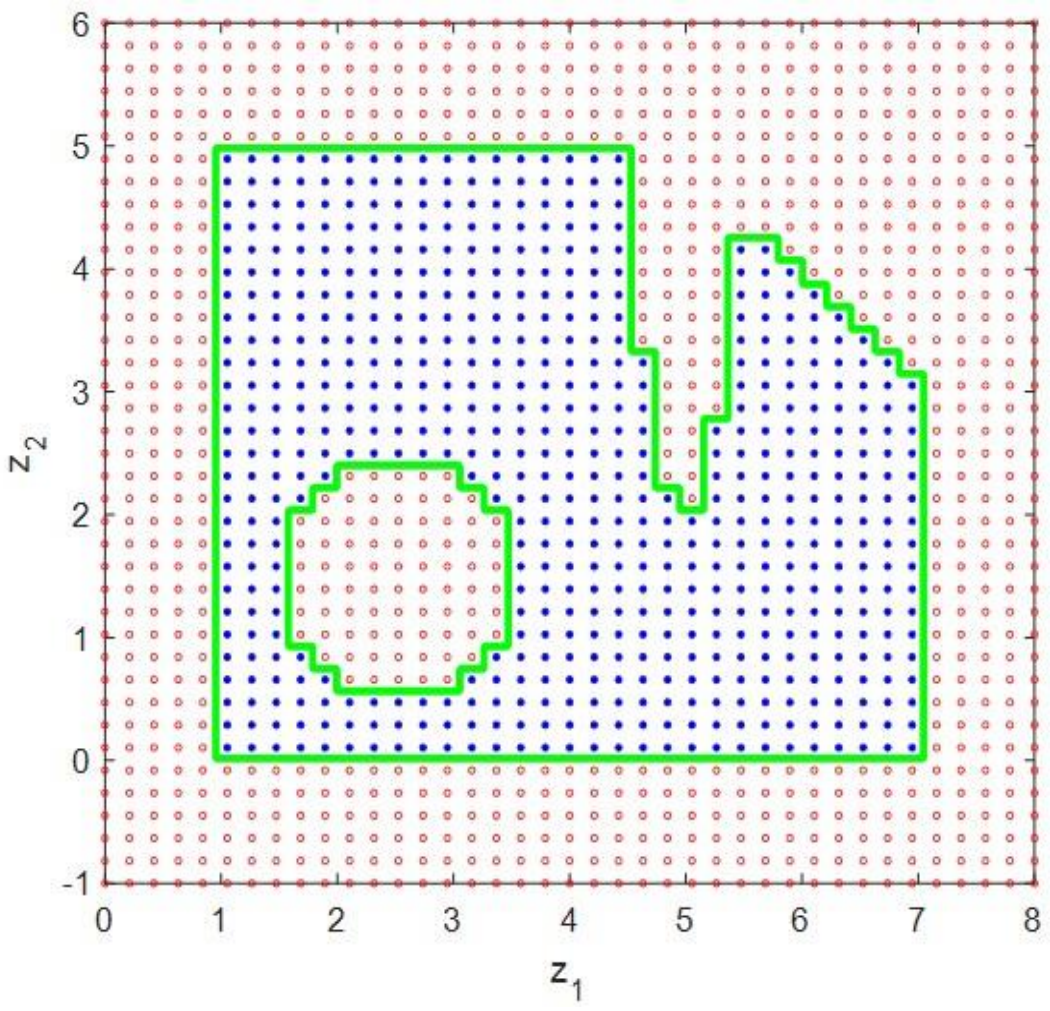

Figure 10.9. Non-parametric periphery identified via CART algorithm with minimum number of samples of one for parent nodes.

11. IMAGE TO CONSTRAINTS: IDENTIFICATION OF CONSTRAINTS FROM IMAGE BOUNDARIES

In this chapter, we will illustrate how to obtain the mathematical equations of the boundaries of the black-and-white (BW) images considering them as "inequality constraints" and using the mathematical techniques introduced in the previous chapters. In fact, with this chapter, we will demonstrate that mathematical expressions (inequality constraint(s) that describe the image boundaries) can be extracted from non-mathematical but visual image objects. In other words, in this chapter our mottos are "*give us a picture and we will give you its mathematical equation*" or "*what you see as beautiful image is actually what we see as beautiful mathematics*".

At the beginning of this chapter, we will mention the generation of the feasible and infeasible points from a BW image. Lastly, we will introduce four sections, each of which includes a single image and the applications of the constraint-identification methods to the image.

Firstly, we will utilize MATLAB's "imread" function to take (read) any digital image (BW or colored, in any industry-standard format, such as the "jpg/jpeg") which can be denoted by a tensor, whose matrices correspond to red, blue, and grey channels, respectively, into MATLAB environment.

The elements of three matrices vary from zero to 255, proportional to the intensity of the pixels. Although these three matrices are different from each other for a colored digital image, we will obtain the three identical matrices, since we will perform only the boundary identification of the BW images in this thesis work.

To reduce the size of the matrices (the size of the image), while retaining the main information of the image, we will use MATLAB's "imresize" function which allows us to decrease the resolution of the image.

To convert an image into a BW (binary) image which is made up of “zeros” and “ones”, we will employ MATLAB’s “im2bw” function. This function replaces the elements of the one of the three identical matrices, which are greater than the certain threshold, with “ones” and the rest with “zeros”. After obtaining the binary matrix, we can determine the indices (co-ordinates) of the “one” and “zero” elements. With these indices of “ones”, we can create the set of feasible points. With the indices of “zeros”, we can form the set of infeasible points.

For instance, let $\mathbf{T} \in \mathbb{R}^{m \times n \times 3}$ be a tensor obtained after the utilization of the MATLAB’s “imresize” function. We will obtain the complete set of all points as $\mathbf{Z} \in \mathbb{R}^{mn \times 2}$ when the rest of the above-mentioned procedure is applied to this tensor. Additionally, we will bring the values of the coordinates of the points into the desired range by using a scale factor so that the points in $\mathbf{Z}$ vary between zero and one, approximately.

Lastly, it should be highlighted that the information on sampling, optimization, and the detailed results, as presented in the previous chapters, will not appear in this chapter in order to keep the focus only on the “image-to-constraint(s)” conversion.

11.1. Image to Approximate and Exact Constraints: A Droplet Image

In this section, we will use the BW image of a droplet for the identification of its boundaries as inequality constraints. We will present two cases for the droplet image. The first one will be the approximate identification of constraints of the droplet with three form-free constitutive constraints, while the second one will be the exact identification of constraint of droplet with a single higher-order polynomial constraint (see Chapter 7 and Chapter 8 for the detailed explanation of such constitutive constraints and their utilization in the context of this thesis work).

Figure 11.1 provides the BW image of the droplet that will be processed for the generation of the sets of feasible and infeasible points to identify the mathematical equation(s) of its boundary as inequality constraint(s).

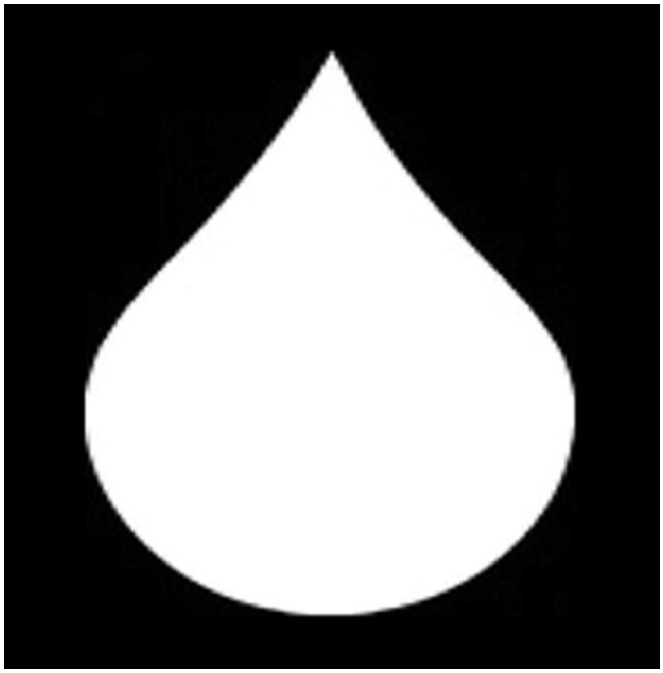

Figure 11.1. BW image of the droplet.

For the first case of this section (the approximate identification), we will use two constitutive linear-interaction constraints (see Equation (7.8) in Chapter 7) together with the constitutive pure-quadratic constraint (see Equation (7.9) in Chapter 7). Thus, the number of optimization decision variables becomes $|\boldsymbol{\theta}| = 13$. Additionally, DE optimizer is employed to solve this image-to-constraint problem. The norm of the parameters is not used in the objective function.

The form-free constitutive inequality constraints identified are given explicitly by the following equations:

$$\hat{g}_1(\boldsymbol{\theta}_1^{\mathrm{I}}) = \quad -1.145 + 0.870z_1 + 0.495z_2 + 0.894z_1z_2, \tag{11.1a}$$

$$\hat{g}_2(\boldsymbol{\theta}_2^{\mathrm{I}}) = \quad -0.004 - 0.016z_1 + 0.013z_2 - 0.002z_1z_2, \tag{11.1b}$$

$$\hat{g}_3(\boldsymbol{\theta}^{\mathrm{PQ}}) = \quad +0.122 - 0.467z_1 - 0.350z_2 + 0.572z_1^2 + 0.353z_2^2. \tag{11.1c}$$

Figure 11.2 presents the aggregated constraint, $\mathbb{C}(\mathbf{P})$, by the light green and the identified individual constitutive constraints by the claret red and cyan for the constitutive linear-interaction constraints, and by the purple circle for the constitutive pure quadratic constraint, over the region formed by the boundary sets of feasible (blue) and infeasible

(red) points generated through processing of the BW image of the droplet. It is possible to observe from Figure 11.2 that the aggregated constraint leads to some violations at the boundary. We obtained the results of the optimization as $J_1 = 0.037$ and $J_2 = 20$, with 13 feasible points out of 162 and 7 infeasible points out of 467. These values also imply the existence of violation of some points. Thus, we obtained the constitutive constraints describing the approximate boundary of the droplet.

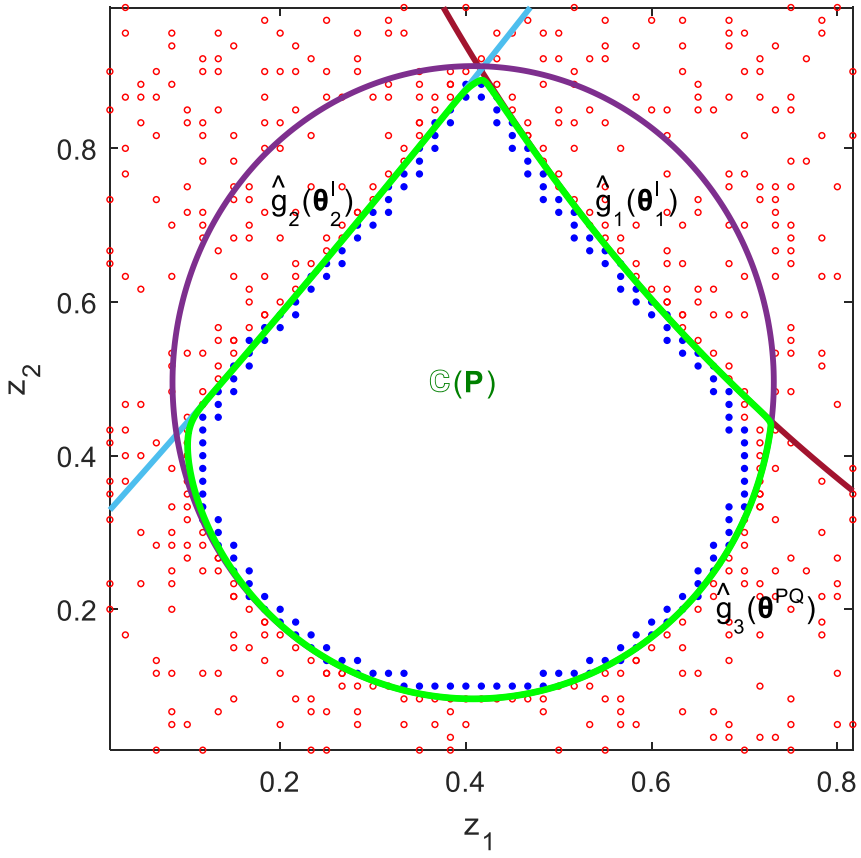

Figure 11.2. Identified approximate constitutive constraints and aggregated constraint superimposed with the boundary sets of feasible and infeasible points for the droplet image.

For the second case of this section (the exact identification), we will use a single third-order polynomial constraint which had explicitly been given in Chapter 8 with the Table 8.7. Thus, the number of optimization decision variables becomes $|\boldsymbol{\theta}| = 16$. Additionally, CMA-ES optimizer is employed to solve this image-to-constraint problem. The norm of the parameters is not used in the objective function.

The form-free 3rd-order polynomial constraint identified is given explicitly by the following equation:

$$\begin{aligned}\hat{g}(\boldsymbol{\theta}) = \; & 6.73 - 21.84z_1 + 36.49z_1^2 - 9.09z_1^3 - 6.72z_2 - 85.57z_1z_2 \\ & - 138.47z_1^2z_2 + 313.65z_1^3z_2 - 56.39z_2^2 + 431.25z_1z_2^2 \\ & + 111.33z_1^2z_2^2 - 848.55z_1^3z_2^2 + 113.42z_2^3 - 579.72z_1z_2^3 \\ & + 229.04z_1^2z_2^3 + 657.03z_1^3z_2^3.\end{aligned} \tag{11.2}$$

Figure 11.3 presents this constraint, $\hat{g}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour) by the light green over the region formed by the boundary sets of feasible (blue) and infeasible (red) points generated via processing the BW image of the droplet. It is possible to observe from Figure 11.3 that the identified constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. The zero value of J, obtained after the optimization, implies this as well. Thus, we obtain the exact 3rd-order polynomial constraint exactly describing the boundary of the droplet.

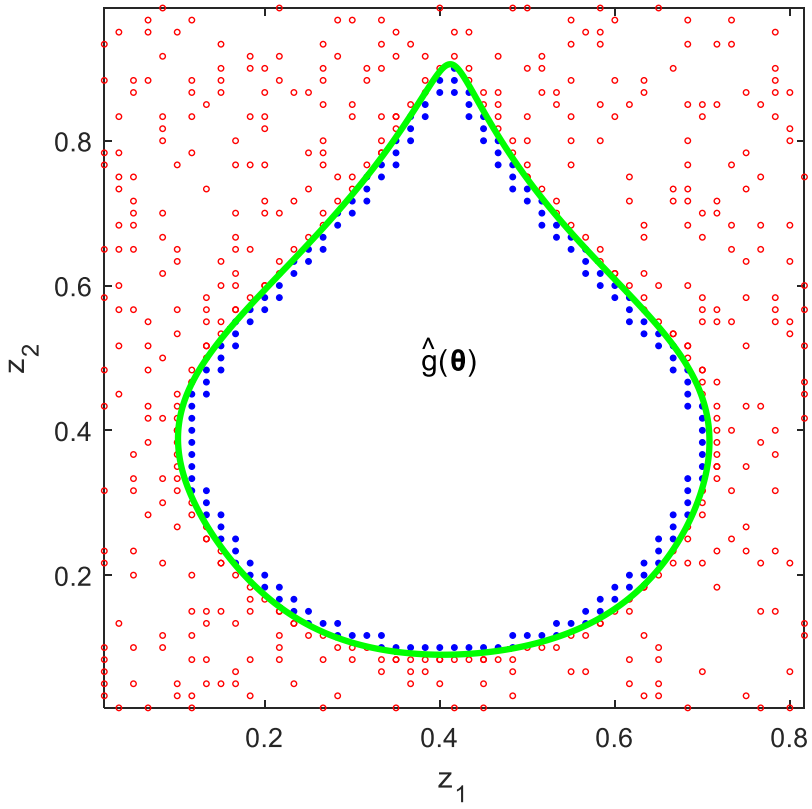

Figure 11.3. Identified optimal single 3rd-order polynomial constraint superimposed with the boundary sets of feasible and infeasible points for the droplet image.

With the form-free constitutives, including two interaction and one pure-quadratic constraints, our algorithm achieved to approximately identify the inequality constraints which lies on the boundary, approximately separating the feasible region from infeasible one. On the other hand, with the single 3rd-order polynomial constraint we completely separated the regions from each other. In other words, our algorithm gave us the one of the true mathematical equations for the boundary of the BW image of the droplet.

Although we can achieve the exact equation of the boundary by the deployment of the single 3rd-order polynomial constraint, physical interpretation of the boundary of the image is impossible since the only the zero contour of the constraint identifies the boundary of the image.

On the other hand, approximate three constitutive constraints can individually be interpreted and these constitutive constraints themselves can be plotted as seen in Figure 11.2. Furthermore, three approximate constitutive constraints require 13 parameters whereas the exact 3rd-order polynomial constraint requires 16 parameters.

11.2. Image to Constraints: A V-Shaped Cup Image

In this section, we will use the BW image of the V-shaped cup for the identification of its boundaries as an inequality constraint. We will present one case for the V-cup image by employing a single higher-order polynomial constraint (see Chapter 7 and Chapter 8 for the detailed explanation of such constitutive constraints and their utilization in the context of this thesis work).

Figure 11.4 provides the BW image of the V-shaped cup that will be processed for the generation of the sets of feasible and infeasible points to identify the mathematical equation of its boundaries as single inequality constraint.

For this image, we will use a single seventh-order polynomial constraint which had been discussed in Chapter 7 and Chapter 8. Thus, the number of optimization decision variables becomes $|\boldsymbol{\theta}| = 64$. Additionally, CMA-ES optimizer is employed to solve this

image-to-constraint problem. The norm of the parameters is not used in the objective function.

Figure 11.4. BW image of the V-shaped cup.

The form-free 7th-order polynomial constraint identified is not explicitly given here because it has $|\boldsymbol{\theta}| = 64$ parameters.

Figure 11.5 presents this constraint, $\hat{g}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour) by the light green over the region formed by the boundary sets of feasible (blue) and infeasible (red) points generated via processing the BW image of the V-shaped cup.

It is possible to observe from Figure 11.5 that the identified constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. The zero value of J, obtained after the optimization, implies this as well. Thus, we obtained a single 7th-order polynomial constraint exactly describing the boundaries of the V-cup image.

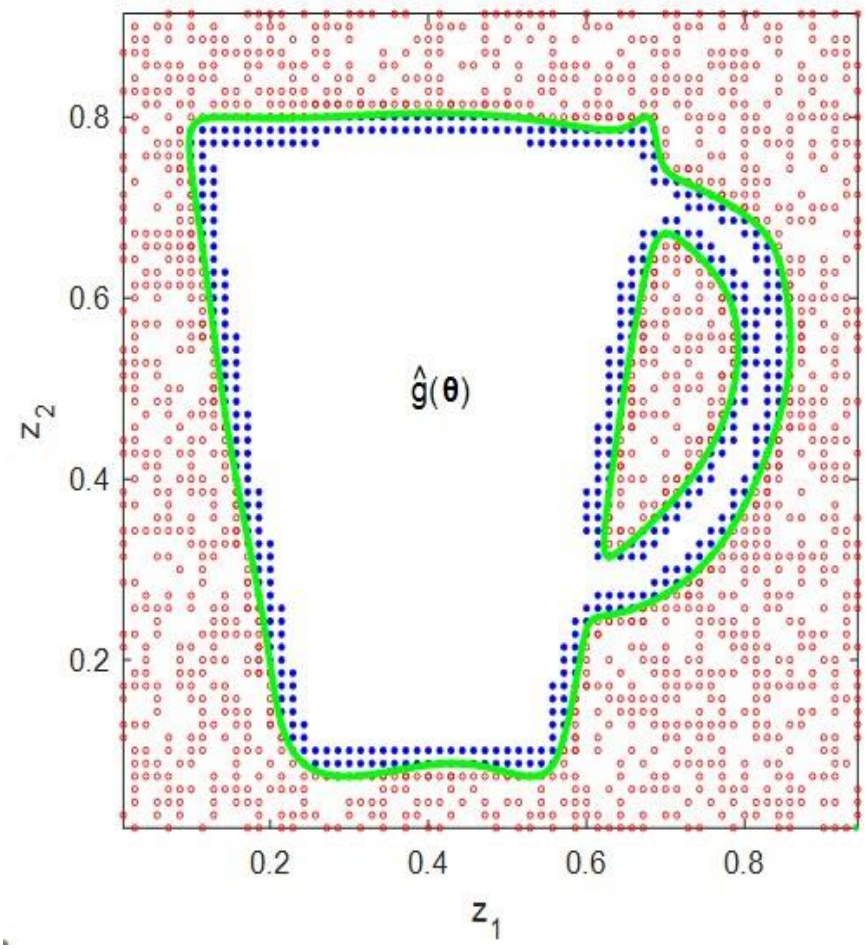

Figure 11.5. Identified optimal single 7th-order polynomial constraint superimposed with the boundary sets of feasible and infeasible points for the V-shaped cup image.

11.3. Image to Constraints: A Disjoint-Regions Image

In this section, we will use the BW image of the disjoint regions (a collection of convex and nonconvex image objects) for the identification of its boundaries as a single inequality constraint. We will present one case for the disjoint-regions image by employing a NN with single output equipped with ReLU activation functions (see Chapter 9 for the detailed explanation of such neural constraints and their utilization in the context of this thesis work).

Figure 11.6 provides the BW image of the disjoint regions that will be processed for the generation of the sets of feasible and infeasible points to identify the mathematical equation of its boundaries as single inequality constraint.

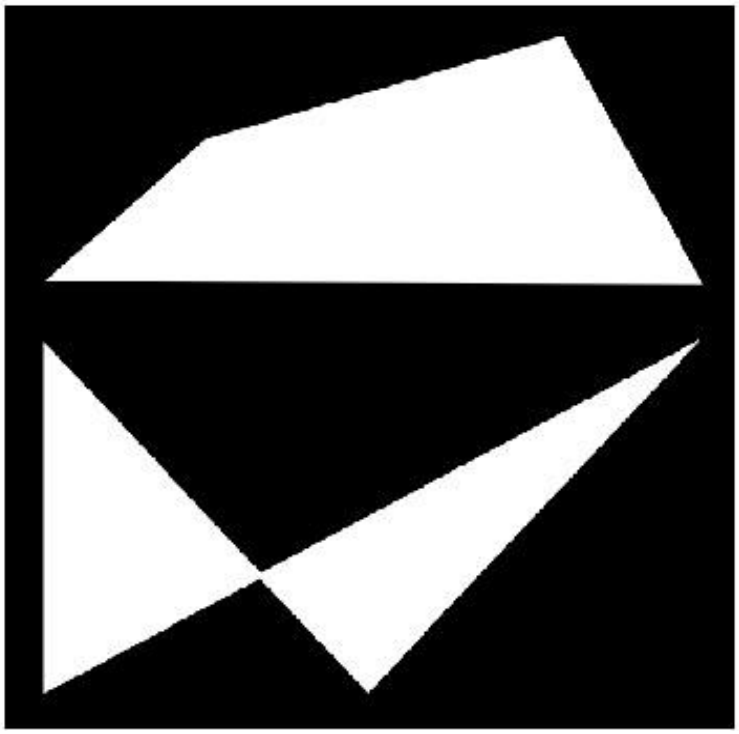

Figure 11.6. BW image of the disjoint regions.

NN topology used for this image includes 50 nodes (neurons) in the hidden layer and single neuron in the output layer, i.e., $\mathrm{N}^{\mathcal{H}} = 50$ and $\mathrm{N}^{\mathcal{O}} = 1$, and thus, the number of optimization decision variables becomes $|\boldsymbol{\theta}| = 201$. We will employ ReLU and linear activation function in the hidden and output neurons, respectively (see Section 9.1 for the detailed explanation of these activation functions and our suggestions about what type of regions they can be used for). Additionally, CMA-ES optimizer is employed to solve this image-to-constraint problem. The norm of the parameters is not used in the objective function.

Figure 11.7 presents this constraint, $\hat{\mathrm{g}}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour) by the light green over the region formed by the boundary sets of feasible (blue) and infeasible (red) points generated via processing the BW image of the disjoint regions. It is possible to observe from Figure 11.7 that the identified neural constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. The zero value of J, obtained after the optimization, implies this as well. Thus, we obtained a single neural constraint that successfully identifies the boundary of the BW image of the disjoint regions as an implicit (neural) function.

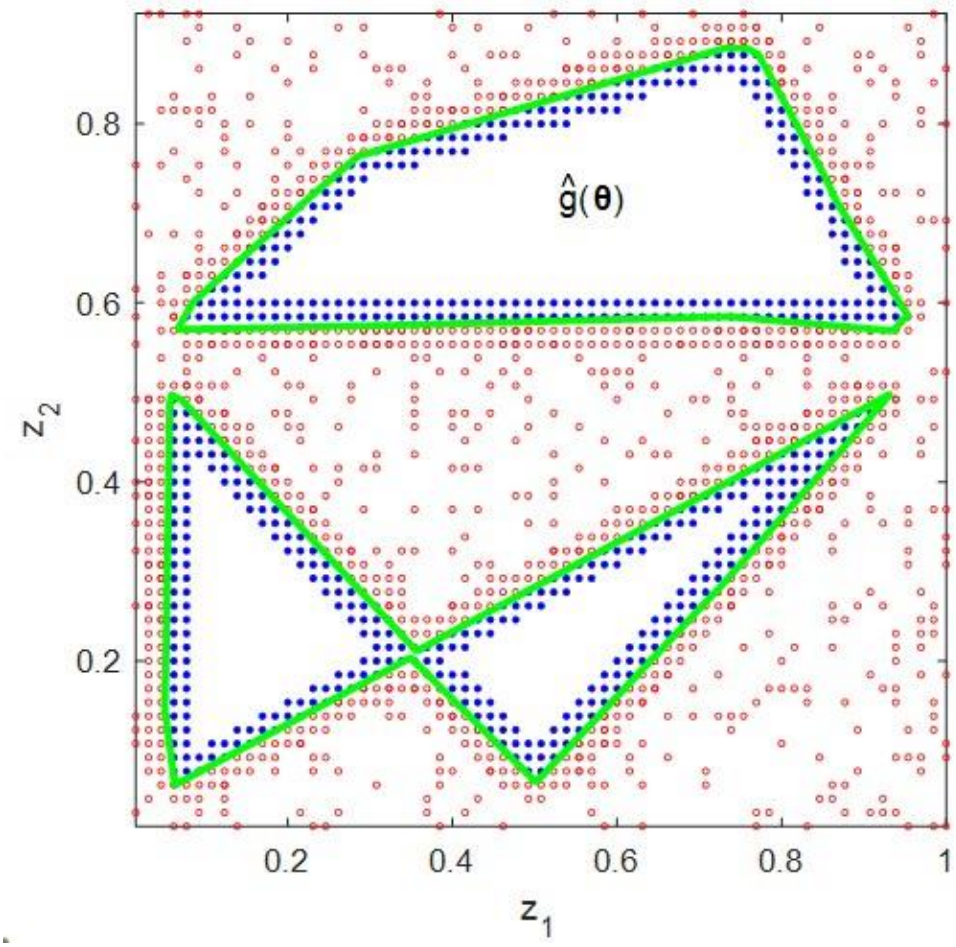

Figure 11.7. Optimal single neural constitutive constraint identified via Neural Network, superimposed with the boundary sets of feasible and infeasible points for the disjoint-regions image.

11.4. Image to Constraints: A Human Image

In the last section of this chapter, we will use a BW human image for the identification of its boundaries as a single inequality constraint. We will present one case for the human image by employing an ELM with single output equipped with unorthodox activation functions (see Chapter 9 for the detailed explanation of such neural constraints and their utilization in the context of this thesis work).

Figure 11.8 provides the BW human image that will be processed for the generation of the sets of feasible and infeasible points to identify the mathematical equation of its boundaries as single inequality constraint.

Figure 11.8. BW human image.

ELM topology used for this image includes 80 nodes (neurons) in the hidden layer and single neuron in the output layer, i.e., $\mathrm{N}^{\mathcal{H}} = 80$ and $\mathrm{N}^{\mathcal{O}} = 1$, and the output bias is used. Therefore, the number of optimization decision variables becomes $|\boldsymbol{\theta}| = 81$. We will employ unorthodox and linear activation function in the hidden and output neurons, respectively (see Section 9.5 for the detailed explanation of these activation functions). Additionally, CMA-ES optimizer is employed to solve this image-to-constraint problem. The norm of the parameters is not used in the objective function.

Figure 11.9 presents this constraint, $\hat{\mathrm{g}}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour) by the light green over the region formed by the boundary sets of feasible (blue) and infeasible (red) points generated via processing the BW human image. It is possible to observe from Figure 11.9 that the identified neural constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. The zero value of J, obtained after the optimization, implies this as well. Thus, we obtained a single neural constraint that successfully identifies the boundary of the BW human image as an implicit (neural) function.

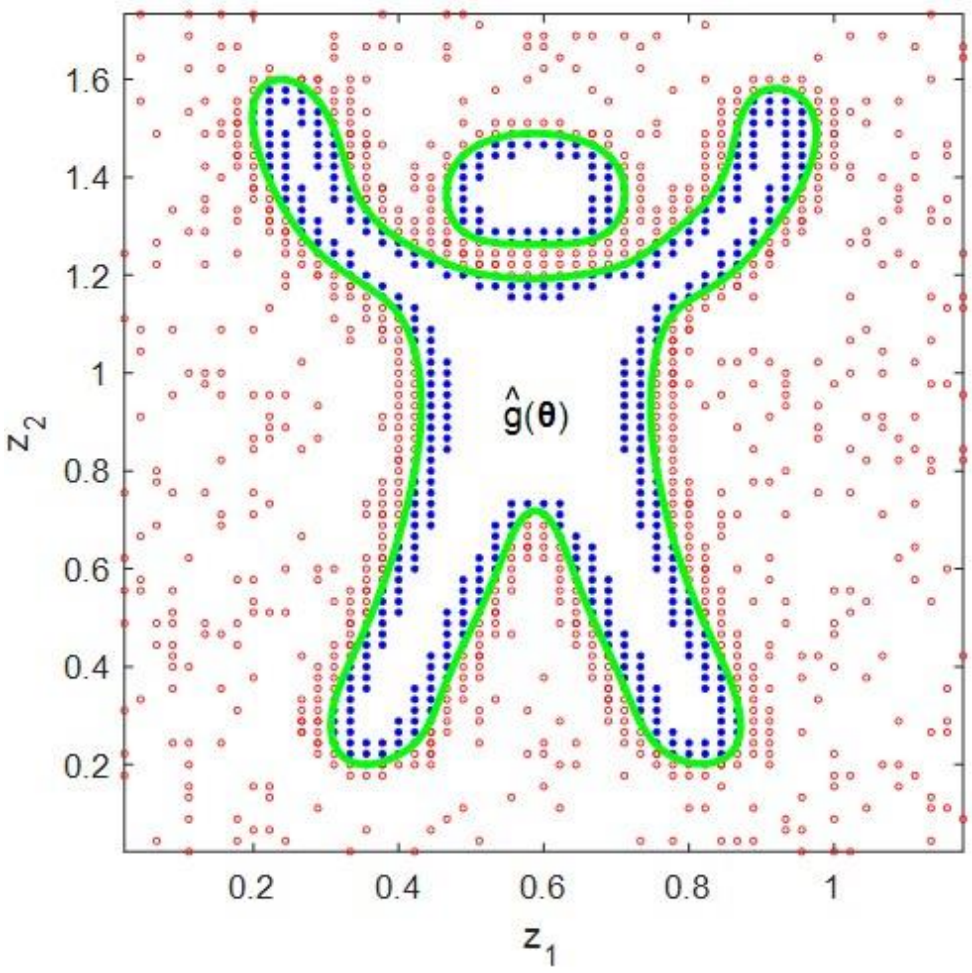

Figure 11.9. Optimal single neural constitutive constraint identified via Extreme Learning Machine, superimposed with the boundary sets of feasible and infeasible points for the human image.

12. SOME CHEMICAL ENGINEERING APPLICATIONS OF CONSTRAINT IDENTIFICATION

In this chapter, we will apply the constraint identification, introduced in the previous chapters of this thesis work, to some Chemical Engineering (ChE) examples. At the onset of this chapter, we will state how to pose optimization problems for chemical plants (or plant units) in the general form and how to reshape them in order to make them suitable for the constraint-identification task. After this brief explanation, we will begin introducing ChE examples with Section 12.1 that will attempt to identify the constraints (the feasible-operation region) of a Continuous Stirred-Tank Reactor (CSTR), and then we will continue with Section 12.2 which provides an example of the identification of the boundary of the vapor-liquid phase-equilibrium diagram, and with Section 12.3 which provides the same example under uncertainty (with measurement errors). Lastly, we will complete this chapter, actually all topics of this thesis work, with the constraint identification of the Williams-Otto Plant which will be presented in Section 12.4.

In general, chemical plant optimization can be described as in the following equations:

$$\min_{\boldsymbol{\beta},\mathbf{x}} \mathrm{f}(\boldsymbol{\beta},\mathbf{x}), \tag{12.1a}$$

s. t.

$$\mathbf{h}(\boldsymbol{\beta},\mathbf{x}) = \mathbf{0}, \tag{12.1b}$$

$$\mathbf{g}(\boldsymbol{\beta},\mathbf{x}) \leq \mathbf{0}, \tag{12.1c}$$

$$\boldsymbol{\beta}^{\mathrm{L}} \leq \boldsymbol{\beta} \leq \boldsymbol{\beta}^{\mathrm{U}}, \tag{12.1d}$$

where $\mathrm{f}(\boldsymbol{\beta},\mathbf{x})$ indicates the objective function, $\mathbf{h}(\boldsymbol{\beta},\mathbf{x})$ is the vector of equality constraints that includes n equations, and $\mathbf{g}(\boldsymbol{\beta},\mathbf{x})$ is the vector of m inequalities. Additionally, $\boldsymbol{\beta}$ and $\mathbf{x}$ indicate the vector of process parameters (uncertain variables) and the vector of n state variables, respectively. The process parameters are also bounded as $\boldsymbol{\beta}^{\mathrm{L}} \leq \boldsymbol{\beta} \leq \boldsymbol{\beta}^{\mathrm{U}}$ where $\boldsymbol{\beta}^{\mathrm{U}}$ and $\boldsymbol{\beta}^{\mathrm{L}}$ imply the upper- and lower-bound vectors, respectively.

For the identification of the feasible region, the presence of equality constraints (if there is any) creates problem. For instance, the existence of only one equality constraint

reduces the feasible-operation region of a chemical plant into a line for a two-dimensional problem. Thus, it is impossible to talk about the feasible-operation region when equality constraint exists, and it cannot be aggregated together with the inequalities. However, in theory, it is possible to eliminate the equalities by elimination of the state variables in terms of the parameters, $\boldsymbol{\beta}$, when the number of the state variables equals to that of the equality constraints (n equations, n unknowns). After such exclusion of the equality constraints, chemical plant optimization problem can be expressed with the following equations, where $f^*(\boldsymbol{\beta})$ and $\mathbf{g}^*(\boldsymbol{\beta})$ are the updated functions after elimination of the state variables:

$$\min_{\boldsymbol{\beta}} f^*(\boldsymbol{\beta}), \tag{12.2a}$$

s. t.

$$\mathbf{g}^*(\boldsymbol{\beta}) \leq \mathbf{0}, \tag{12.2b}$$

$$\boldsymbol{\beta}^L \leq \boldsymbol{\beta} \leq \boldsymbol{\beta}^U. \tag{12.2c}$$

Through this chapter, equalities will be eliminated in this way, if they are present, and the aim is to identify feasible-operation region formed by $\mathbf{g}^*(\boldsymbol{\beta})$, $\boldsymbol{\beta}^L$ and $\boldsymbol{\beta}^U$. In conjunction of the formulations used in the previous chapters of this thesis, $\boldsymbol{\beta}$ corresponds to sample points in axis directions, i.e., $\mathbf{z}$, thus $\boldsymbol{\beta} \in \mathbb{R}^{1\times D}$ is the row vector sampled N times in all D dimensions. N^f of these sample points correspond to the feasible points and N^i of them correspond to the infeasible points.

It should be mentioned that this elimination of the equality constraints may not always be possible, e.g., if the equalities are nonlinear such that multiple solutions of state variables arise. This problem is outside the scope of this thesis work.

12.1. A CSTR Example

In this section, we will identify the feasible-operation region of a two-step reaction, occurring in CSTR (Zhao et al., 2022). The mechanism of the reaction is given below together with the equations of the reaction rates (r_1 and r_2), where k_1 and k_1 are the reaction rate constants set to 0.31051 mol^{-1} L s^{-1} and 0.026650 s^{-1}, respectively:

$$A + B \overset{k_1}{\Rightarrow} C, \qquad r_1 = k_1 C_A C_B, \tag{12.3a}$$

$$C \overset{k_2}{\Rightarrow} D + E, \qquad r_2 = k_2 C_C. \tag{12.3b}$$

For this reaction, there exist two process parameters which are the residence time (β_1) and the ratio of the concentration of B to A (β_2), and these process parameters are bounded as:

$$0 \leq \beta_1 \leq 550, \tag{12.4a}$$

$$0 \leq \beta_2 \leq 6. \tag{12.4b}$$

The mass-balance equations for the CSTR reaction, included in the vector of equality constraints, i.e., $\mathbf{h}(\boldsymbol{\beta}, \mathbf{x}) = \mathbf{0}$, are given with the following equations, where the state variables are $\mathbf{x} = [C_A \quad C_B \quad C_C \quad C_D \quad C_E]$ and the initial concentrations are $[C_A^0 \quad C_B^0 \quad C_C^0 \quad C_D^0 \quad C_E^0] = [0.53 \quad 0.53\beta_2 \quad 0 \quad 0 \quad 0]$:

$$h_1: \quad C_A^0 - C_A + \beta_1(-r_1) = 0, \tag{12.5a}$$

$$h_2: \quad C_B^0 - C_B + \beta_1(-r_1) = 0, \tag{12.5b}$$

$$h_3: \quad C_C^0 - C_C + \beta_1(r_1 - r_2) = 0, \tag{12.5c}$$

$$h_4: \quad C_D^0 - C_D + \beta_1(r_2) = 0, \tag{12.5d}$$

$$h_5: \quad C_E^0 - C_E + \beta_1(r_2) = 0. \tag{12.5e}$$

The minimum yield of species D and the minimum ratio of species D to the amount of unreacted species, which are the inequality constraints of the problem, $g_j(\boldsymbol{\beta}, \mathbf{x}) \leq 0$, are as follows:

$$g_1: \quad -\frac{C_D}{C_A^0 - C_A} + 0.9 \leq 0, \tag{12.6a}$$

$$g_2: \quad -\frac{C_D}{C_A + C_B + C_C} + 0.2 \leq 0. \tag{12.6b}$$

As mentioned at the onset of this chapter, our aim is to identify the feasible operation region formed by $\mathbf{g}^*(\boldsymbol{\beta})$ (see Equation (12.2)), obtained via the elimination of the state variables in $\mathbf{g}(\boldsymbol{\beta}, \mathbf{x})$ (see Equation (12.1)) using the equality constraints (Equation (12.5)). It should be observed that the process parameters do not explicitly appear in the inequality constraints (Equation (12.6)), but in the equality constraints (Equation (12.5)). Thus, the state variables can be re-defined as $\mathbf{x} = \mathbf{x}(\boldsymbol{\beta})$ and $\mathbf{g}(\boldsymbol{\beta}, \mathbf{x})$ can be replaced with $\mathbf{g}\big(\mathbf{x}(\boldsymbol{\beta})\big)$.

Additionally, the first reaction rate brings non-linearity into equality constraints in terms of the state variables and the nonlinearity makes it impossible to explicitly eliminate the state variables. Thus, rather than eliminating the state variables algebraically, we find the values of four state variables (from C_A to C_D) by employing MATLAB's fsolve function which solves four equality constraints (h_1 to h_4) for four unknowns (C_A, C_B, C_C, C_D).

We will employ the constitutive bound constraints together with single constitutive linear-interaction constraint for the identification of feasible-operation region. Thus, the number of the constitutive constraints and the number of parameters become $N^{Con} = 5$ and $|\boldsymbol{\theta}| = 8$, respectively.

CMA-ES optimizer is used, and the norm of the parameters is the part of the overall objective function with $\mu = 10^{-4}$. SLHS is employed to generate sample points by adjusting the value of the number of sample points as $N = 2500$. The procedure of the boundary-zone formation is applied to this problem with $p = 2$ (Chapter 3). The value of rho is set to $\rho = 500$ for the KS function that aggregates the five constitutive constraints.

The constitutive bound constraints together with single constitutive linear-interaction constraint, identified for the CSTR problem, are given explicitly by the following equations:

$$\hat{g}_1\left(\boldsymbol{\theta}_1^{UB}\right) = -549.860 + \beta_1, \tag{12.7a}$$

$$\hat{g}_2\left(\boldsymbol{\theta}_2^{UB}\right) = -5.583 + \beta_2, \tag{12.7b}$$

$$\hat{g}_3(\boldsymbol{\theta}_1^{LB}) = +337.700 - \beta_1, \tag{12.7c}$$

$$\hat{g}_4(\boldsymbol{\theta}_2^{LB}) = +0.182 - \beta_2, \tag{12.7d}$$

$$\hat{g}_5(\boldsymbol{\theta}^{I}) = 10^{-4} \times (-0.000 - 1.943\beta_1 + 16.427\beta_2 + 0.316\beta_1\beta_2). \tag{12.7e}$$

Figure 12.1 provides the aggregated constraint, $\mathbb{C}(\mathbf{P})$, by the light green and the identified individual constitutive constraints by the claret red, purple, cyan, yellow lines (constitutive bound constraints), and orange curve (constitutive linear-interaction constraint) over the region formed by the boundary sets of feasible (blue) and infeasible (red) points generated. It is possible to observe from Figure 12.1 that the aggregated

constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible boundary points. We obtained $J_1 = J_2 = 0$ as the results of optimization.

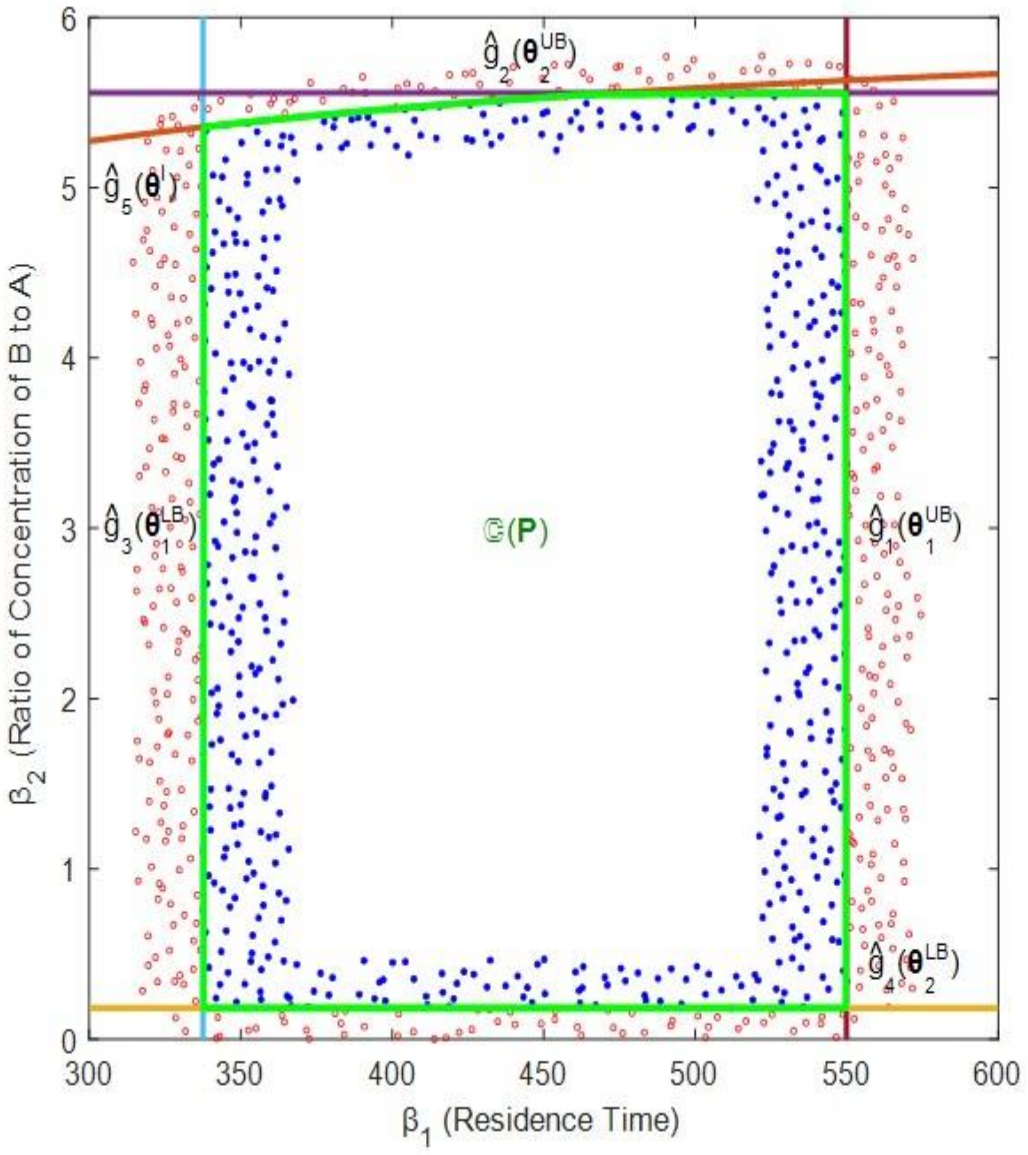

Figure 12.1. Identified optimal constitutive constraints and aggregated constraint superimposed with the boundary sets of feasible and infeasible points for the CSTR-reaction problem.

By the deployment of the constitutive bound constraints together with single constitutive linear-interaction constraint, our algorithm successfully identified the feasible operation-region of the CSTR problem without scaling optimization decision variables

between a certain range, i.e., between zero and one. Additionally, it actively used all of the constitutive constraints so that none of the constitutive constraints became redundant.

It should be noted that, in this example, for the first time in this thesis, we solved a both the inequality- and equality-constrained problem and proposed the solution of set of nonlinear algebraic equations during the sample-generation phase of our approach. This methodology is feasible only if the total degrees of freedom considering the equality constraints is greater than or equal to two. For this problem, we had six variables; four state variables (C_A, C_B, C_C, C_D) and two process parameters (β_1, β_2), and four equality constraints, leading to a degrees of freedom of two, and thus, we were able to plot and identify the feasible region over the two-dimensional plane. If, for any problem, the total degrees of freedom is only one (n variables and n−1 equality constraints), then one degrees of freedom will lead to a one-dimensional problem for which the feasible region cannot be sampled since it is a curve or line, not a region.

12.2. Binary Phase Envelope Example

In this chapter, we will give a constraint-identification example on how to extract the mathematical equation (enclosing inequality constraint) of the phase-equilibrium envelope of ethane-benzene mixture using a single polynomial constraint. Additionally, here, we will do something interesting with regard to the generation of the feasible and infeasible points such that we will generate the sets of the feasible and infeasible points based only on the set of experimental data points. Thus, for the first time in this thesis, rather than employing actual inequality constraints to generate feasible and infeasible points, we are using the experimental data points directly.

We simulate the experimental data of the phase-equilibrium diagram of the ethane-benzene mixture using the Soave-Redlich Kwong equation of state by calculating the ethane-benzene equilibrium mole fractions as a function of pressure (P) at constant temperature (T). The sample points generated thus can be considered as experimental data, for our purposes, since the equation of state uses the mixture parameters fitted to the actual data at $T = 160\ ℃$ which are depicted in Figure 12.2, where the blue points denotes the points on the saturated-liquid curve (the bubble-point curve where the first bubble of gas

appears upon pressure decrease) and the red ones represent the points on the saturated-vapor curve (the dew-point curve where the first drop of liquid ethane appears upon pressure increase). In this figure, the left-hand side of the saturated-liquid curve and the right-hand side of the saturated-vapor curve are the single-phase zones where only the liquid or gaseous states exists, respectively. Inside of the envelope is the two-phase region where the liquid and gas phases coexist. The x-axis of blue points corresponds to the mole fraction of the liquid ethane (x_{ethane}) and the x-axis of red points corresponds to the mole fraction of the gaseous ethane (y_{ethane}).

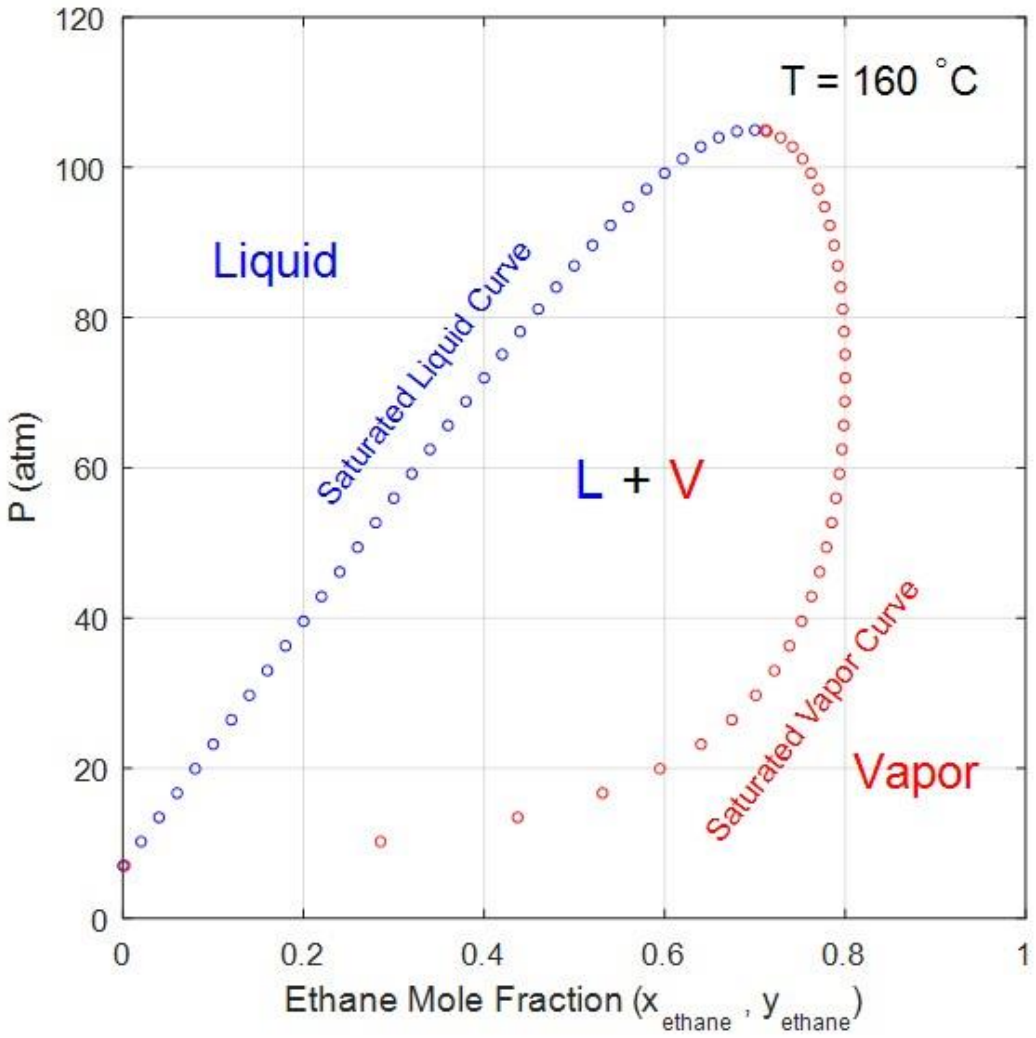

Figure 12.2. Ethane-benzene phase-equilibrium diagram.

Computed (fitted) ethane-benzene phase-equilibrium data at 160 °C depicted in the above figure are given in Table 12.1 below.

Table 12.1. Computed ethane-benzene phase-equilibrium data at 160 °C.

x_{ethane}	y_{ethane}	P (atm)
0.0001	0.0020466	7.0285
0.020	0.285	10.221
0.040	0.437	13.442
0.060	0.531	16.676
0.080	0.594	19.921
0.100	0.640	23.177
0.120	0.674	26.441
0.140	0.700	29.714
0.160	0.721	32.993
0.180	0.738	36.277
0.200	0.751	39.563
0.220	0.762	42.851
0.240	0.771	46.137
0.260	0.779	49.418
0.280	0.785	52.693
0.300	0.789	55.957
0.320	0.793	59.208
0.340	0.796	62.440
0.360	0.798	65.649
0.380	0.799	68.830
0.400	0.800	71.976
0.420	0.799	75.080
0.440	0.798	78.135
0.460	0.797	81.131
0.480	0.794	84.057
0.500	0.791	86.900
0.520	0.787	89.645
0.540	0.782	92.275
0.560	0.777	94.767
0.580	0.770	97.095
0.600	0.762	99.228
0.620	0.752	101.13
0.640	0.741	102.74
0.660	0.728	103.99
0.680	0.712	104.80
0.700	0.711	104.98

In this example, there are no actual inequality constraints; we only have the set of experimental data points forming the binary phase envelope given in Table 12.1 above. However, our algorithm requires the existence of the sets of feasible and infeasible points to identify the mathematical equation of the binary phase envelope. Therefore, for the first

time, we will employ a method of generation of the sets of feasible and infeasible points based solely on the set of the experimental data points.

Firstly, we take the experimental data points, i.e., x_{ethane}, y_{ethane}, and P as the variables by considering the saturated-liquid and saturated-vapor curves separately as $x_{ethane} = f_1(P)$ and $y_{ethane} = f_2(P)$, respectively (see Figure 12.2 and observe the blue and red branches of the envelope). Then, we generate two-dimensional sample points (pressure P and ethane mole fraction Z without distinguishing them as x or y) by using one of the sampling methods introduced in this thesis (Chapter 3).

Since the generated sampling points do not necessarily correspond to data values given in Table 12.1, we use MATLAB's interp1 function to individually interpolate the $x_{ethane} = f_1(P)$ and $y_{ethane} = f_2(P)$ curves using the values given in Table 12.1. This gives the values of Z^{LC} and Z^{VC}, corresponding to the sampled value of P, respectively.

Lastly, we introduce two inequality constraints; the first one which classifies the sample points generated to the right of the P-interpolated saturated-liquid curve as feasible (Equation (12.8a)) and the other which classifies the sample points generated to the left of the P-interpolated saturated-vapor curve as feasible (Equation (12.8b)).

,We use the variable Z for the ethane mole fraction sample points, without distinguishing gas or liquid state, and this variable is taken as the horizontal axis of the feasible-region diagram. We use the variable P as the vertical axis of the feasible-region diagram. These inequalities, which are not inherited, but created for the system of the ethane-benzene mixture, are given explicitly by:

$$g_1: \quad +Z - Z^{LC} \leq 0, \tag{12.8a}$$

$$g_2: \quad -Z + Z^{VC} \leq 0. \tag{12.8b}$$

It should be noted that the variable P does not explicitly appear in these inequalities and all the terms in these equations describe mole fraction of ethane. However, the numerical values of Z^{LC} and Z^{VC} depend on the interpolated (from Table 12.1) data at the sampled P values, thus P implicitly appear in these equations.

We will use a single 3rd-order polynomial constraint to identify the feasible (two-phase) region. Thus, the number of parameters becomes $|\boldsymbol{\theta}| = 16$. CMA-ES optimizer is used, and the norm of the parameters is not utilized.

SLHS is employed to generate sample points (pressure P and ethane mole fractions Z) by adjusting the value of the number of sample points as $\mathrm{N} = 7500$. The procedure of the boundary-zone formation is not applied to this problem (Chapter 3).

The single 3rd-order polynomial constraint, identified for the binary phase envelope, is given explicitly by the following equation:

$$\begin{aligned}\hat{\mathrm{g}}(\boldsymbol{\theta}) = {} & +0.138 + 0.882\mathrm{Z} - 0.029\mathrm{Z}^2 + 3.002\mathrm{Z}^3 - 0.014\mathrm{P} \\ & + 0.696\mathrm{PZ} + 7.425\mathrm{PZ}^2 - 0.067\mathrm{PZ}^3 - 0.004\mathrm{P}^2 \\ & - 0.159\mathrm{P}^2\mathrm{Z} - 0.902\mathrm{P}^2\mathrm{Z}^2 + 1.098\mathrm{P}^2\mathrm{Z}^3 + 0.000\mathrm{P}^3 \\ & + 0.006\mathrm{P}^3\mathrm{Z} - 0.007\mathrm{P}^3\mathrm{Z}^2 + 0.001\mathrm{P}^3\mathrm{Z}^3.\end{aligned} \tag{12.9}$$

Figure 12.3 presents this 3rd-order polynomial constraint, $\hat{\mathrm{g}}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour) by the light green over the region formed by the complete sets of feasible (blue, V+L two-phase region) and infeasible (red, either V or L phase regions outside the two-phase envelope) points generated from the set of so-called experimental data points (computes phase equilibrium data) and the inequality constraints created for this problem.

It is possible to observe from Figure 12.3 that the identified constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible points. The zero value of J, obtained after the optimization, implies this as well.

Figure 12.4 provides single 3rd-order identified polynomial constraint superimposed only with the set of experimental data points. It is possible to observe from this figure that the green envelope, denoting the zero-valued contour of the identified polynomial constraint, totally fits the points of the saturated-liquid phase (blue points) and the saturated-vapor phase (red points) branches and thus, fits to the ethane-benzene two-phase envelope.

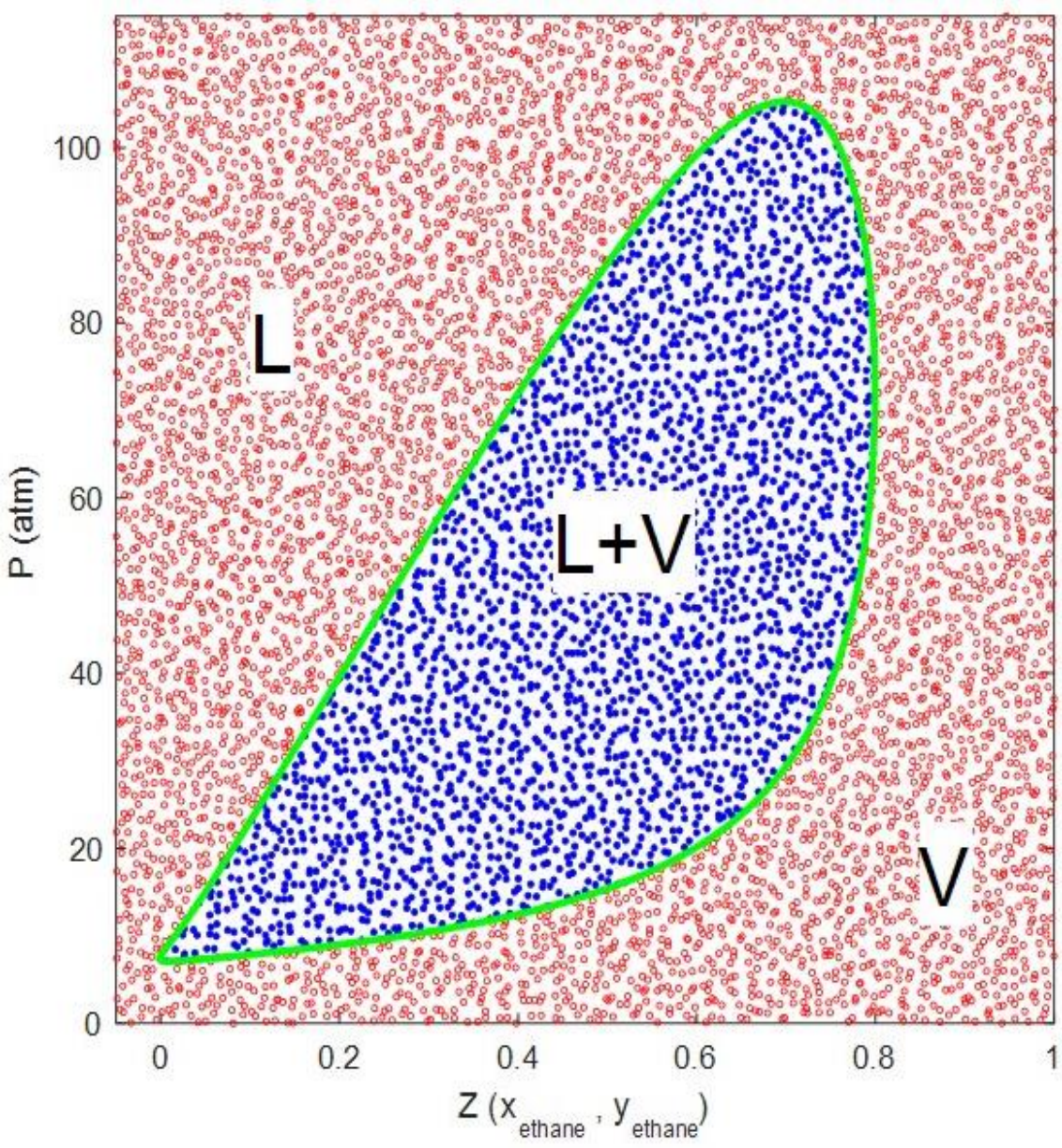

Figure 12.3. Identified optimal single 3rd-order polynomial constraint superimposed with the complete sets of feasible (two-phase) and infeasible (single-phase) points for the ethane-benzene mixture.

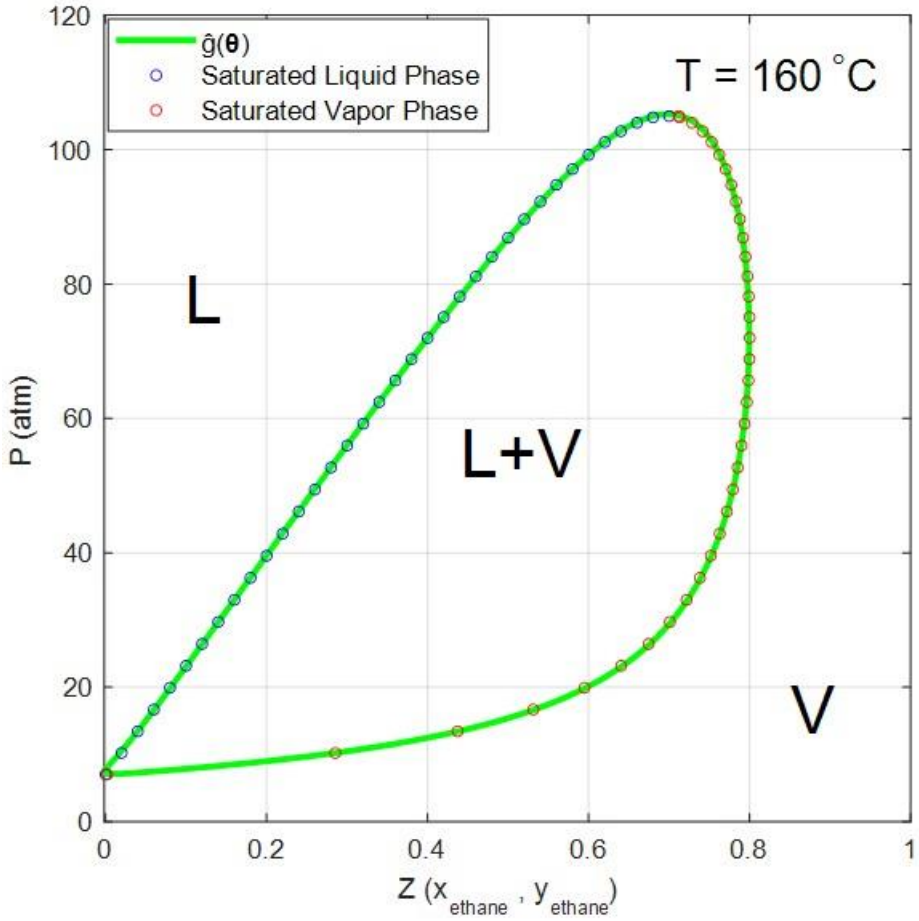

Figure 12.4. Ethane-benzene phase-equilibrium diagram with the identified optimal single 3rd-order polynomial constraint.

12.3. Binary Phase Envelope Example Under Uncertainty

In this section, we, for the first time, will provide an example that shows the feasible-region identification under data uncertainty (in the presence of measurement errors). As in the previous section, we will identify enclosing inequality constraint of the phase-equilibrium envelope of ethane-benzene mixture using a single polynomial constraint. However, the experimental data of ethane-benzene phase-equilibrium, at this time, include some measurement errors (uniform random noise) added to the data provided with Section 12.2.

For this section, we will employ a single 3rd-order polynomial constraint to identify the feasible (two-phase) region. Thus, the number of parameters becomes $|\boldsymbol{\theta}| = 16$. CMA-

ES optimizer is used, and the norm of the parameters is not utilized. SLHS is employed to generate sample points (pressure P and ethane mole fractions Z) by adjusting the value of the number of sample points as $N = 5000$ (Chapter 3). As elucidated in Section 12.2, we label generated sample points as feasible or infeasible. Additionally, the boundary-zone formation procedure is not applied to this problem.

The single 3rd-order polynomial constraint identified for the binary phase envelope under measurement errors is given explicitly by the following equation:

$$\begin{aligned}\hat{g}(\boldsymbol{\theta}) = {} & +0.059 + 0.209Z + 0.365Z^2 - 0.598Z^3 - 0.012P \\ & - 0.041PZ + 0.097PZ^2 - 0.039PZ^3 + 0.001P^2 \\ & - 0.002P^2Z + 0.002P^2Z^2 - 0.001P^2Z^3 - 0.000P^3 \\ & + 0.000P^3Z - 0.000P^3Z^2 + 0.000P^3Z^3.\end{aligned} \tag{12.10}$$

Figure 12.5 presents this 3rd-order polynomial constraint, $\hat{g}(\boldsymbol{\theta})$, (i.e., actually its zero-valued contour) by the light green over the region formed by the complete sets of feasible (blue, V+L two-phase region) and infeasible (red, either V or L phase regions outside the two-phase envelope) points generated via the set of experimental data points containing uncertainty (generated by adding random noise to the computed phase equilibrium data) and the inequality constraints present in Section 12.2. It is possible to observe from Figure 12.5 that the identified constraint leads to some violations at the boundary, and thus, violates some of the feasible and infeasible points (71 out of 5000).

Figure 12.6 provides single 3rd-order identified polynomial constraint superimposed with the set of uncertain experimental data points. It is possible to observe from this figure that the green envelope, representing the zero-valued contour of the identified polynomial constraint, does not completely fit the points of the saturated-liquid phase (blue points) and the saturated-vapor phase (red points) branches formed by the experimental data containing noise. For comparison purposes, we superimposed the identified constraint in the previous section (noise-free two-phase region). By comparing the zero-valued contours of the two identified constraints, it is possible to say that polynomial constraint identified under uncertainty very closely approximates the true two-phase region.

Since the experimental data of the phase-equilibrium diagram has some noise, it is not expected that our algorithm classifies all the feasible and infeasible points totally correctly. In other words, the noise existing in the data causes our algorithm to classify some sample points as feasible or infeasible incorrectly. This situation has already been observed from Figure 12.5. As a result of this situation, our algorithm is able to reach optimal 3rd-order polynomial constraint, corresponding to the actual binary-phase envelope, at least, the most approximate one. The value of J greater than zero indicates that the identified polynomial constraint shows some violations at the boundary.

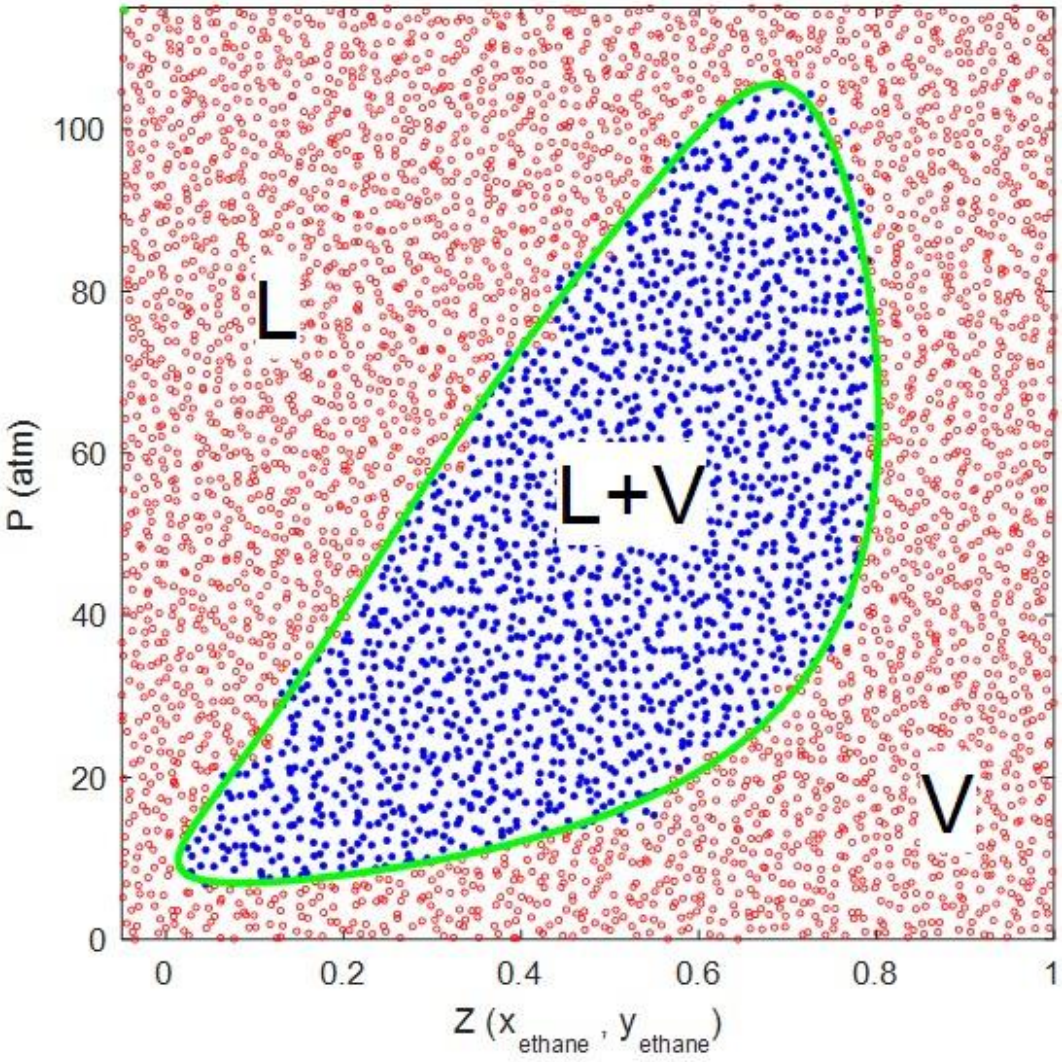

Figure 12.5. Identified optimal single 3rd-order polynomial constraint superimposed with the complete sets of feasible (two-phase) and infeasible (single-phase) points including uncertainty for the ethane-benzene mixture.

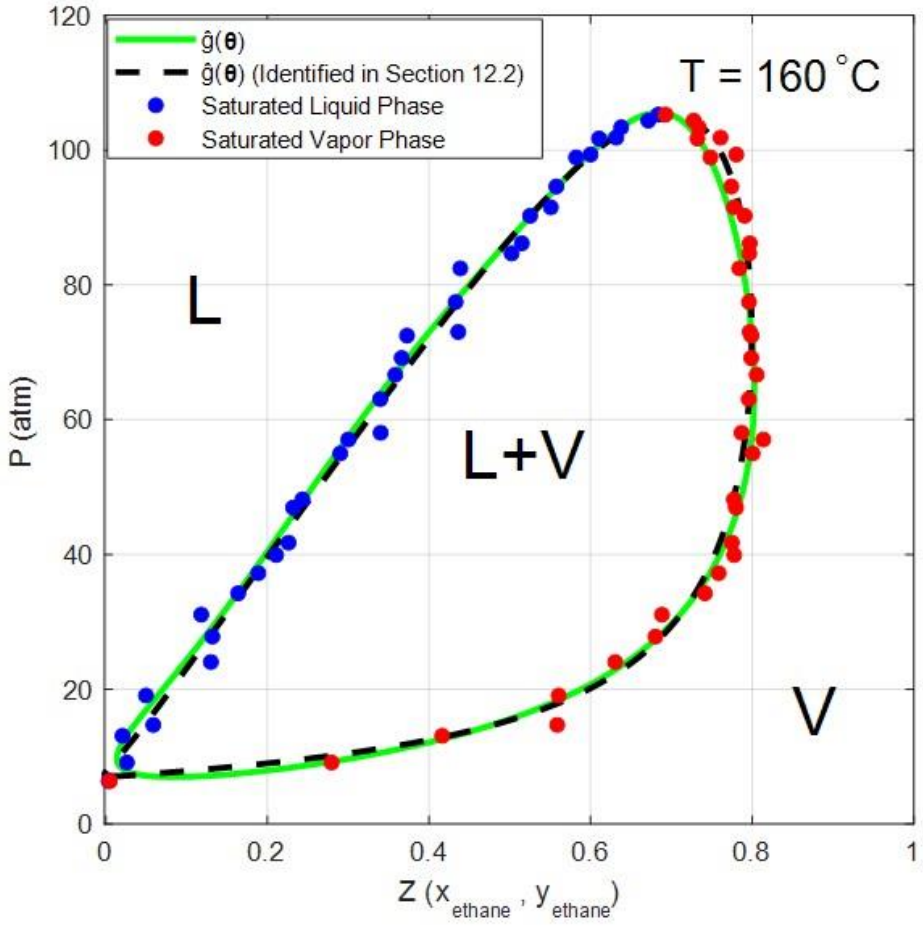

Figure 12.6. Ethane-benzene phase-equilibrium data including uncertainty superimposed with the identified optimal single 3rd-order polynomial constraint.

In cases where the uncertainty exists in the experimental data, the choice of the constitutive constraint has a significant impact on the results. It is important to employ a conservative constitutive constraint (a parsimonious model with regard to number of parameters) which is not complex such that it does not capture the noise and fits and filters the experimental data containing measurement errors. For instance, a single very high-order polynomial constitutive constraint or neural constitutive constraint with excessive number of neurons can fit even the measurement errors and perfectly separates all the feasible and infeasible points containing added noise, i.e., perfect identification of jagged boundaries is theoretically possible.

In this example, with the use of relatively low-order (3rd-order) constitutive constraint, our identification algorithm acted as a regression method (regressor) that

minimizes the sum of the squared errors, well filtered out the noise, and thus, captured the almost true (measurement-error free) two-phase region.

12.4. The Williams-Otto Plant Example

In this last section of Chapter 12, we will provide the constraint identification of the Williams-Otto Plant (WOP) as an example by employing the form-specific constitutive constraints.

WOP consists of a Continuous-Stirred Tank Reactor (CSTR), decanter, distillation column and splitter, and contains a recycle stream that creates numerical-solution- and optimization-related difficulties. The simplified (adopted in the literature) flowsheet for the WOP is given in Figure 12.7 (Williams and Otto, 1960; Rijckaert and Martens, 1974; Edgar et al., 2001).

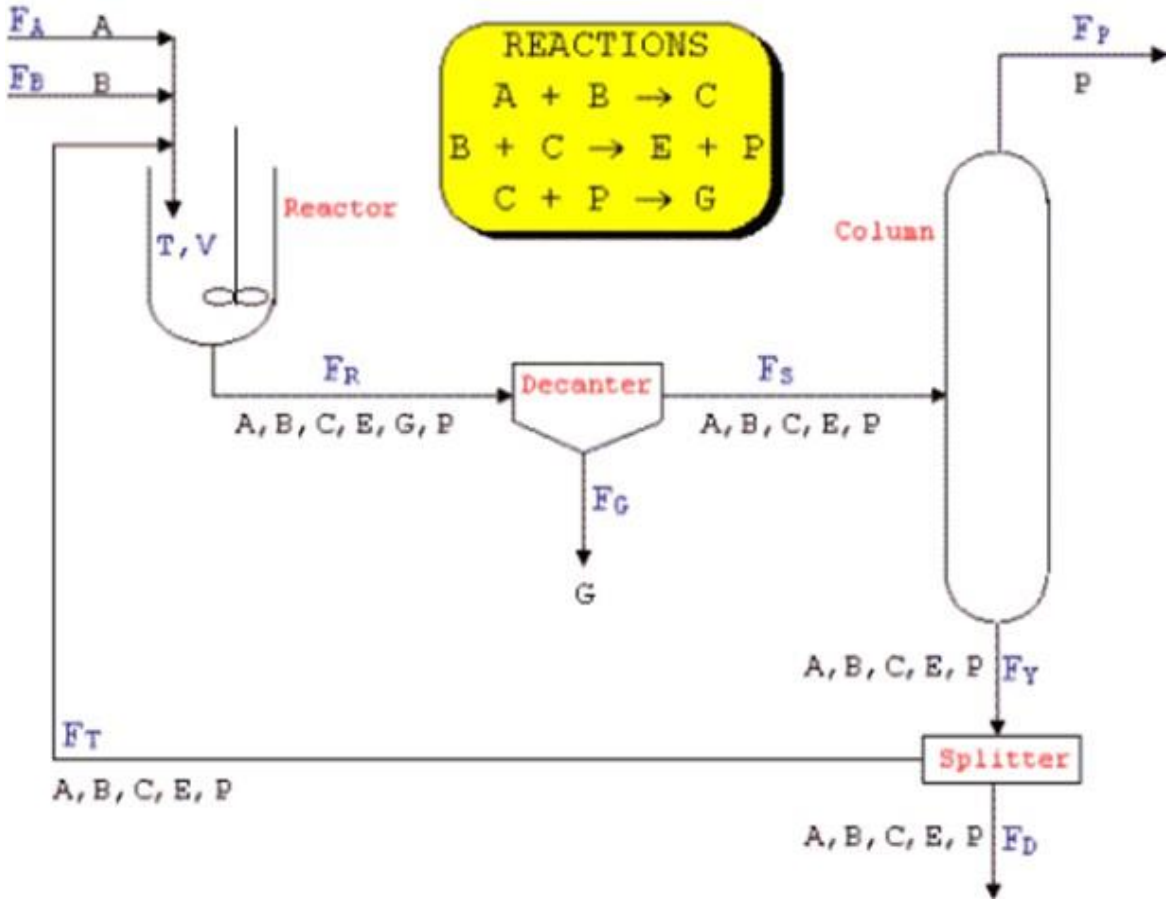

Figure 12.7. Flowsheet of the Williams-Otto plant.

The components A and B separately enters the CSTR with mass flowrates F_A and F_B (lb/h), respectively. The three reactions given in the above figure occur in the CSTR. As a result of these three reactions, intermediate product C, main product P, waste product G and species E are produced. The mixture, taken from the outlet stream of CSTR and formed by four products, C, E, G, P, and unreacted amount of species A and B, leaves the CSTR with total mass flowrate of F_R (lb/h). This mixture joins the decanter where waste product G is completely separated from the mixture containing six species. After the decanter that removes the waste product G from the mixture with mass flowrate of F_G, the top stream of the decanter enters the distillation column to obtain main product as pure P from the top with a mass flowrate of F_P (lb/h). Additionally, small amount of main product (10 wt% of mass flowrate of species E in the stream fed to the column) appears in the bottoms stream of the column due to the formation of an azeotrope. Some amount of the mixture, taken from the bottoms stream of the column (stream Y), is returned to CSTR with total mass flowrate F_T (lb/h) and the rest is utilized as fuel (F_D lb/h). It should be also highlighted that F_{ki} denotes the mass flowrate of species i at stream k represented in Figure 12.7.

After this brief explanation of the WOP, we will give the mass-balance model of this plant, which is identical to the models adopted in the literature for various optimization-related studies. The model considers only isothermal operation. This simple model of the WOP well serves the purpose of providing a "plant-wide constraint identification" example.

The reaction rates are given together with reaction rate constants in the following equations:

$$k_i = a_i e^{\frac{-b_i}{T_R}} \left(\frac{V\rho_d}{F_R^2}\right); \quad i = 1,2,3, \tag{12.11a}$$

$$r_1 = k_1 F_{RA} F_{RB}, \tag{12.11b}$$

$$r_2 = k_2 F_{RB} F_{RC}, \tag{12.11c}$$

$$r_3 = k_3 F_{RC} F_{RP}, \tag{12.11d}$$

where the density of the mixture is $\rho_d = 50$ lb/ft^3, the volume of CSTR is $V = 30$ ft^3 and the constants are $[a_1 \quad a_2 \quad a_3] = 10^{12} \times [0.0059755 \quad 2.5962 \quad 9628.3]$ and $[b_1 \quad b_2 \quad b_3] = 10^3 \times [12 \quad 15 \quad 20]$.

The mass-balance equations around the CSTR are written as follows:

$$F_A + F_{TA} - F_{RA} - r_1 = 0, \quad (12.12a)$$

$$F_B + F_{TB} - F_{RB} - r_1 - r_2 = 0, \quad (12.12b)$$

$$F_{TC} - F_{RC} + 2r_1 - 2r_2 - r_3 = 0, \quad (12.12c)$$

$$F_{TE} - F_{RE} + 2r_2 = 0, \quad (12.12d)$$

$$F_{TG} - F_{RG} + 1.5r_3 = 0, \quad (12.12e)$$

$$F_{TP} - F_{RP} + r_2 - 0.5r_3 = 0. \quad (12.12f)$$

The mass-balance equations around the decanter, proving perfect separation between species G and others, are given as:

$$F_{Si} = F_{Ri}; \quad i = A, B, C, E, P, \quad (12.13a)$$

$$F_{SG} = 0, \quad (12.13b)$$

$$F_{Gi} = 0; \quad i = A, B, C, E, P, \quad (12.13c)$$

$$F_{GG} = F_{RG}. \quad (12.13d)$$

The distillation column is where the pure species P is taken from the top. However, some amount of species P, 10 wt% of the mass flowrate of species E, remains in the mixture taken from the bottoms of the column due to formation of an azeotrope. This is modelled by the mass-balance equations around the distillation column as given in the following equations:

$$F_{Yi} = F_{Si}; \quad i = A, B, C, E, \quad (12.14a)$$

$$F_{Pi} = 0; \quad i = A, B, C, E, G, \quad (12.14b)$$

$$F_{PP} = F_{SP} - 0.1F_{SE}, \quad (12.14c)$$

$$F_{YP} = F_{SP} - F_{PP} = 0.1F_{SE}, \quad (12.14d)$$

$$F_{GG} = F_{RG}. \quad (12.14e)$$

For the splitter, which divides its feed stream into two output streams with the split fraction $\alpha = 0.1$, the mass-balance equations are given in the following equations:

$$F_{Ti} = \alpha F_{Yi}; \quad i = A, B, C, E, P, \quad (12.15a)$$

$$F_{Di} = (1 - \alpha)F_{Yi}; \quad i = A, B, C, E, P. \quad (12.15b)$$

In addition to the above-given process constraints (mass balances can be considered as equality constraints), we have the operation constraints (inequality constraints) which

determines the feasible-operation region of the WOP. The mass flowrate of the species B in the feed stream of the CSTR, F_B, can be adjusted to 68000 lb/h at most (Equation (12.16a)). The temperature of the CSTR, T_R, should be kept between 610 °R and 675 °R (Equation (12.16b) and Equation (12.16c)). Additionally, the ratio (mass fraction) of the species A in the outlet stream of the CSTR, i.e., $x_A = F_{RA}/F_R$, is desired to be 18.5 percent at most (Equation (12.16d)). The ratio of species G (waste product), i.e., $x_G = F_{RG}/F_R$, is desired to be 4.5 percent at most (Equation (12.16e)). The mass flowrate of pure species P in the top stream of the distillation column, F_P, is bounded from above with the value of 5000 lb/h as well (Equation (12.16f)). The value of F_A is set to 13000 lb/h. These inequalities are all mathematically expressed as:

$$g_1: \quad +F_B - 65000 \leq 0, \tag{12.16a}$$

$$g_2: \quad -T_R + 610 \leq 0, \tag{12.16b}$$

$$g_3: \quad +T_R - 675 \leq 0, \tag{12.16c}$$

$$g_4: \quad +x_A - 0.185 \leq 0, \tag{12.16d}$$

$$g_5: \quad +x_G - 0.045 \leq 0, \tag{12.16e}$$

$$g_6: \quad +F_P - 5000 \leq 0. \tag{12.16f}$$

The uncertain variables (process parameters), to be used as process parameters for the identification of two-dimensional feasible-operation region of the WOP. are selected to be the mass flowrate of B into the CSTR and the reactor temperature, i.e., F_B, and T_R, respectively. In other words, two-dimensional sample points (the mass flowrate of B in the feed stream of the CSTR F_B and the reactor temperature T_R) are generated via a sampling method introduced in Chapter 3 and the values of other parameters (state variables) are found by solving the mass-balance equations around CSTR (Equation (12.12a-f)) and the other mass-balance equations (Equation (12.13-15)), as simultaneous algebraic equations via MATLAB's fsolve function, at each sampling instant.

The SLHS method is used to generate sample points with $N = 1500$. The procedure of the formation of the boundary zone is not applied (Chapter 3). The value of rho is set to $\rho = 500$ for the KS function that aggregates constitutive constraints (Chapter 4). We will employ constitutive bound constraints (see Equation (7.6) in Chapter 7) together with two constitutive linear-interaction constraints (see Equation (7.8) in Chapter 7). Thus, the number of the optimization decision variables becomes $|\boldsymbol{\theta}| = 12$. Additionally, CMA-ES

optimizer is employed to solve the WOP problem. The norm of the parameters is used in the objective function by $\mu = 10^{-4}$.

The form-specific constitutive bound constraints and linear-interaction constraints identified are given explicitly in the following equations:

$$\hat{g}_1\left(\boldsymbol{\theta}_1^{\mathrm{UB}}\right) = -0.928 + \Theta_1, \tag{12.17a}$$

$$\hat{g}_2\left(\boldsymbol{\theta}_2^{\mathrm{UB}}\right) = -0.978 + \Theta_2, \tag{12.17b}$$

$$\hat{g}_3\left(\boldsymbol{\theta}_1^{\mathrm{LB}}\right) = -0.000 - \Theta_1, \tag{12.17c}$$

$$\hat{g}_4\left(\boldsymbol{\theta}_2^{\mathrm{LB}}\right) = +0.884 - \Theta_2, \tag{12.17d}$$

$$\hat{g}_5\left(\boldsymbol{\theta}_1^{\mathrm{I}}\right) = -0.502 + 0.000\Theta_1 + 0.568\Theta_2 - 0.078\Theta_1\Theta_2, \tag{12.17e}$$

$$\hat{g}_6\left(\boldsymbol{\theta}_2^{\mathrm{I}}\right) = +0.408 - 0.000\Theta_1 - 0.391\Theta_2 - 0.123\Theta_1\Theta_2, \tag{12.17f}$$

where $\Theta_1 = \mathrm{F_B}/70000$ and $\Theta_2 = \mathrm{T_R}/690$ imply the scaled uncertain variables.

Figure 12.8 provides the aggregated constraint, $\mathbb{C}(\mathbf{P})$ (actually, its zero-valued contour), by the light green and the identified individual constitutive constraints by the claret red, purple and yellow lines (constitutive upper-bound constraint on Θ_1, i.e., $\mathrm{F_B}$, constitutive upper- and lower-bound constraints on Θ_2, i.e., $\mathrm{T_R}$), orange and cyan lines (constitutive linear-interaction constraints) over the region formed by the complete sets of feasible (blue) and infeasible (red) points generated for the WOP problem. It is possible to observe from Figure 12.8 that the aggregated constraint does not lead to any violations at the boundary and thus satisfies all the feasible and infeasible boundary points. We obtained $\mathrm{J}_1 = \mathrm{J}_2 = 0$ as the results of optimization.

By the use of constitutive bound constraints and two constitutive linear-interaction constraints, our algorithm had successfully identified the feasible operation region of the WOP. The constitutive lower-bound constraint on $\mathrm{F_B}$, $\hat{g}_3(\boldsymbol{\theta}_1^{\mathrm{LB}})$, had emerged as redundant, since it had not existed in Figure 12.8. Had the ranges of the axes been enough, it would clearly have appeared in Figure 12.8 as redundant constitutive.

The WOP example could have been solved equally well via using form-free (unspecific) constitutive constraint(s) as well, for example by employing single- or multiple-output NN or ELM.

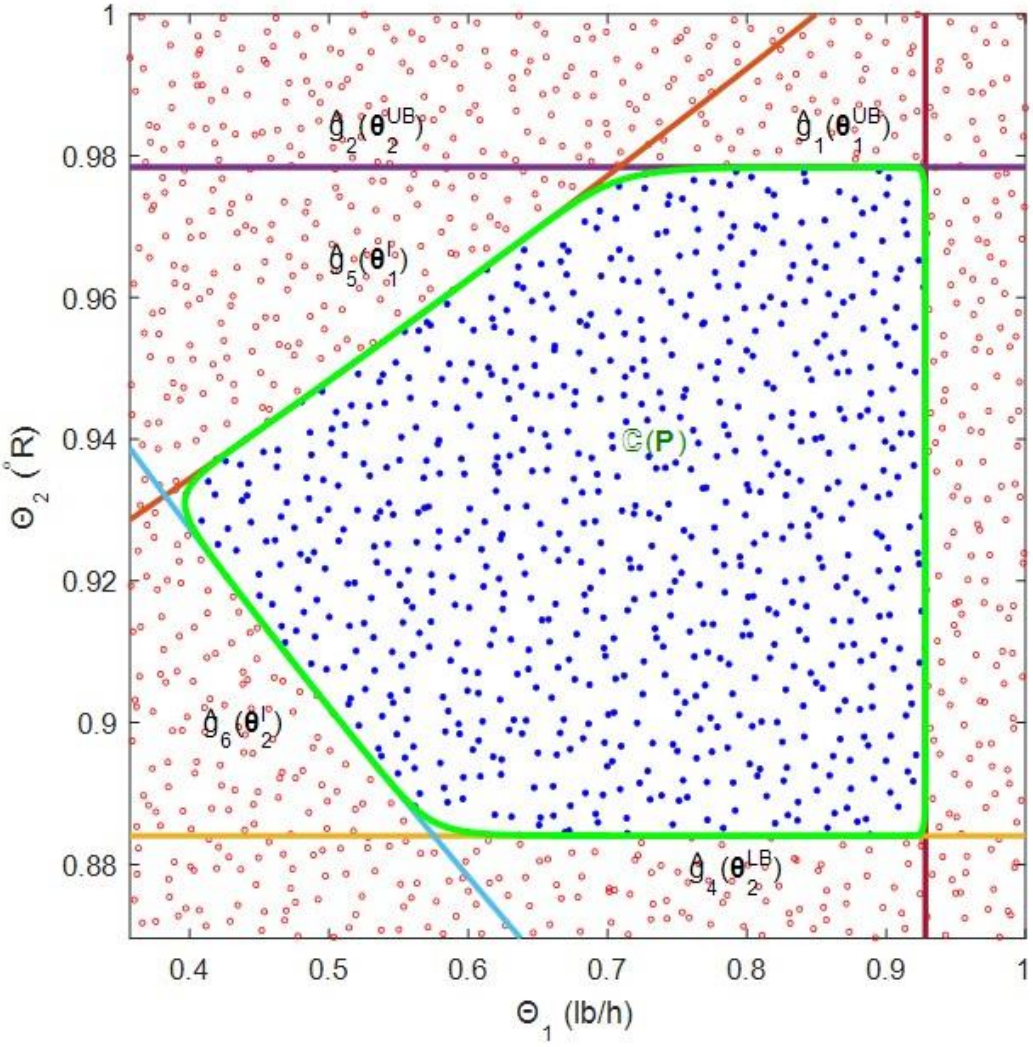

Figure 12.8. Five optimal constitutive constraints identified and aggregated constraint superimposed with the complete sets of feasible and infeasible points for the WOP problem.

13. CONCLUSIONS AND RECOMMENDATIONS

13.1. Conclusions

The ultimate aim of this thesis work, the “data-based constraint identification”, was to create a methodology from scratch to be able to identify several types of D-dimensional feasible and infeasible regions. Towards this aim, firstly we created “a constraint-identification algorithm” by viewing the task of constraint identification from the perspective of optimization. After the creation of its basis, we equipped our algorithm with different novel concepts such as “constitutive constraints”, “form-specific constitutives”, “form-free constitutives”, “neural (machine-learned) constitutives” and “image-based constraints”, all of which were introduced and developed throughout this thesis work for the first time in the literature.

In Chapter 2, we began with the definitions of “feasible/infeasible regions” and the explanation of why we worked on developing a method related to the identification of the feasible/infeasible regions. We continued with the illustrations of convex and nonconvex regions and explained the need for the utilization of nonlinear constitutives (regressor inequality constraints) so as to identify nonconvex regions by depicting two situations. Lastly, we completed the chapter with the illustration of two-feasible regions formed by two individual inequalities.

In Chapter 3, we posed three different sampling methods, Random Sampling (RS), Stratified Latin Hypercube Sampling (SLHS), and Meshwise Sampling (MS), together with their short explanations. We mentioned how to produce the complete, feasible, and infeasible sets step by step. We explicitly exhibited the “set of all sample points” and the “sets of feasible and infeasible points”. To build comprehension about sampling methods in the context of constraint-identification task, we illustrated the sampling of the same feasible region via three sampling methods with six different number of sample points. We thoroughly discussed the advantages and disadvantages of these sampling methods and examined the effect of the cardinality of the set of sample points on the constraint-

identification task over 18 different illustration of the same feasible region. In this chapter, we also proposed the “boundary-zone formation” based on the concept of “percentile of distances” and explained its computational procedure. The “boundary-zone formation” feature provided our algorithm with the competence of reaching optimum parameter values of the constitutive constraints with much less computing times. After the brief explanation of this feature, we examined the method of the boundary-zone formation by sampling regions via SLHS and MS with different number of sample points and different values of the percentile level.

In Chapter 4, we presented the topic of constraint aggregation by focusing on the underestimator and overestimator forms of the Kreisselmeier-Steinhauser (KS) aggregation function. We studied the effect of the value of tightening parameter of KS aggregation function, ρ, by depicting the feasible regions formed by the same set of inequality constraints with different values of ρ. At the closure of this chapter, we explained why the constraint-identification task required such an aggregator function and highlighted the type of KS aggregation function and its numerically-stabilized form utilized throughout the thesis work.

Chapter 5 was devoted to the development of the concept of “form-specific constitutives” by considering specific shapes of inequality constraints such as bound constraints, linear constraints, circular constraints, and ellipsoidal constraints. In this chapter, we utilized form-specific, both “linear-in-parameters” and “nonlinear-in-parameters”, constitutives that were appropriate for the identification of the D-dimensional regions. At the onset of the chapter, we provided details on how the form-specific constitutives are evaluated at the feasible and infeasible points and showed four types of the form-specific constitutives in matrix form. Then, we explained the application of the KS aggregation function to form-specific constitutive inequality constraints, evaluated over the feasible and infeasible points, in order to attain a single aggregated constraint describing the feasible region. By taking the single aggregated constraint into account, for our optimization problem, we developed the overall objective function, which consisted of three sub-objective functions. We thoroughly elucidated what to consider when selecting the parts of the most suitable overall objective function for the constraint-identification task as well. We briefly explained the algorithm of the Differential Evolution (DE)

optimization and the DE computer code which was utilized mostly to solve the constraint-identification examples of this thesis work.

Chapter 6 encompassed the constraint-identification examples with ten different feasible regions, all of which were successfully identified via the form-specific constitutives. Firstly, in this chapter, so as to illustrate that the boundary-zone formation worked as intended, we provided two cases for each of the two feasible regions, formed by two- and three-dimensional circular inequality constraints. Then, we continued with two identification examples of four- and eight-dimensional circular inequality constraints to prove that our algorithm can successfully identify inequality constraints in much higher dimensions as well. Additionally, we illustrated the constraint-identification examples for ellipsoidal inequality constraints and bound constraints, both in 2-D and 3-D. We exemplified the identification of the feasible region formed by linear inequality constraints only in 2-D. The feasible regions of Chapter 6 were all convex except for the last one. As the last example of this chapter, we identified a nonconvex feasible region formed, as opposed to the other feasible regions, by multiple types of inequalities. With Chapter 6, we proved that our boundary-zone formation feature, form-specific constitutive constraints, objective function, and our optimization procedure could be deployed for the purpose of identification of the inequalities forming the region. The basic form of our algorithm successfully identified the feasible and infeasible regions. The SLHS and MS sampling methods were both utilized for the identification examples of Chapter 6.

Chapter 7 was not only devoted to developing the concept of “form-free constitutives” via the “design matrix” approach, but also it accommodated the redesign of the “form-specific constitutive bound constraints” and the “form-specific constitutive linear inequality constraints” via the design-matrix approach. In the form-free concept, the constitutives maintained their existence as “linear-in-parameters” model equations. However, our algorithm gained more flexibility and capability to identify more complex feasible regions, for instance, regions formed by shape-wise mathematically unperceptible multiple types of inequalities. In this chapter, we had provided five design (model) matrices and their creation logic for five types of constitutives, in detail. After this, we illustrated how our algorithm handled the mixture of constitutives with examples which explicitly showed the formation of the “augmented design matrix” and “sparse parameters

matrix". For the sake of unification of this chapter, we provided the procedure of merging the constitutives into a single aggregated constraint and the route to extraction of optimal parameters of constitutives after termination of the optimization. At the closure of Chapter 7, we mentioned further possibilities for the creation of different more complex design matrices, which also gave us the idea of the utilization of the Machine Learning (ML) tools as constitutives.

Chapter 8 included the constraint-identification examples solved by the use of multiple types of constitutive constraints and single high-order polynomial constitutive constraints. The constitutive constraints deployed were all built via the design matrix. In the first five sections of Chapter 8, we provided the constraint-identification examples of five different feasible regions solved with six different utilizations of multiple types of constitutive constraints. The feasible regions presented in these sections were all nonconvex except for the first one. We showed that our algorithm successfully identified the "two-dimensional disjoint infeasible region" and "two-dimensional fragmented disjoint feasible region" by the use of multiple types of constitutive constraints, in Section 8.3 and Section 8.4, respectively. We gave a constraint-identification example of three-dimensional feasible region in Section 8.5 as well. In Section 8.6, we shortly explained how to build single higher-order polynomial constitutive constraint with the design-matrix approach by means of a factorial-design specification. We explicitly illustrated the building blocks of single third-order polynomial constitutive constraint as well. After short explanation of the creation of the single polynomial constitutive constraint, we demonstrated the successful identification of the feasible regions with the single polynomial constitutive constraints not only in Section 8.6, but also in Section 8.7 and Section 8.8. Section 8.7 held the illustration of the successful identification of the regions of Section 8.3 and Section 8.4 by single 5^{th}-order and single 7^{th}-order polynomial constitutive constraints, respectively. We completed Chapter 8 with the concept and illustration of an "approximate identification" of the convex-circular feasible region. In this section, we successfully obtained piecewise-linear approximation (almost perfect octagonal representation) of circular feasible region. Additionally, we proved that the utilization of the constitutive bound constraint together with constitutive linear inequality constraints could ensure the perfection of the octagonal representation of circular feasible region. This approximate constraint-identification

example was solved by CMA-ES optimizer. To sample the regions presented in this chapter, both the SLHS and MS methods were utilized.

Chapter 9 was dedicated to creating the concept of the form-free implicit “neural constitutive constraints” via two ML algorithms. In this chapter, we developed the neural constitutive constraint(s) as nonlinear-in-parameters models by deploying Single Layer Feedforward Neural Networks (SLFNNs). Such neural constitutive constraints furnished our algorithm with endless flexibility and allowed our algorithm to identify much more complex feasible regions with ease without requiring decisions on the mathematical form of the constitutive constraint equations. Then, to capitalize on their approximation capabilities with much less parameters compared to SLFFNs, we introduced the utilization of Extreme Learning Machines (ELM) for the creation of implicit yet more parsimonious nonlinear models that were “linear-in-parameters”, which were also called neural constitutive constraints. Chapter 9 included the constraint-identification examples of two regions via utilization of neural constitutives build via these ML algorithms with three different activation functions as well. In the first four sections of Chapter 9, we provided the constraint-identification examples of the same convex region learned via single-output NN, multiple-output NN, and ELM with ReLU and ReSech activation functions. Within Section 9.1 and Section 9.2 we showed that the best identification of convex regions (especially linearly convex ones) could be obtained with the utilization of the ReLU activation function. Inside Section 9.3 and Section 9.4, we illustrated that it could be possible to achieve the same or comparable constraint-identification ability of NNs with the ELMs. Additionally, in the last three sections of Chapter 9, we provided the constraint-identification examples of the same nonconvex region learned via single-output NN and ELM, multiple-output NN with an unorthodox activation function (newly introduced in this thesis by combining the parametric version of radial-basis activation function with exponential function). For all the constraint-identification examples of Chapter 9, the SLHS method was utilized and these identification examples had been learned with the utilization of both the DE and CMA-ES optimizers.

In Chapter 10, we deployed five classification algorithms; Probabilistic Neural Network (PNN), k-Nearest Neighbours (k-NN), Support Vector Machine (SVM), Gaussian Process Regression (GPR), and Classification and Regression Trees (CART), for the

periphery identification (classification of feasible and infeasible regions) of a nonconvex region sampled via MS method. After the short explanation of these techniques, we scrutinized the effect of the hyperparameters of these classification algorithms on the periphery identification. For the periphery identification of nonconvex region, PNN classification, with the spread of 0.5, gave a nonparametric periphery, which led to violations of some feasible and infeasible points, and thus, failed to represent true feasible and infeasible regions. We attained true representation of the boundary of the nonconvex region with the spread value of 0.1. k-NN classification algorithm failed to successfully classify feasible and infeasible points of nonconvex region with the higher value of its hyperparameter, $k = 17$, while giving a nonparametric periphery satisfying all feasible and infeasible points at the boundary with the lower value of its hyperparameter, $k = 3$. However, it was more likely to achieve true, but highly zigzagged periphery via k-NN classification algorithm for lower values of its hyperparameter. For the periphery identification via SVM classification, we examined the effect of the value of the kernel scale factor on this task. Through two cases employing the kernel scale factor of one and 0.2, we demonstrated that SVM classification was successful at achieving a true nonparametric periphery with the lower kernel scale factor. For the periphery identification of nonconvex region via GPR algorithm, we presented a single case by utilizing linear and "matern 3/2" as basis and covariance-kernel functions of GPR. In this case, GPR algorithm successfully classified all feasible and infeasible points and gave true nonparametric periphery nearly identical to one obtained via SVM classification with the kernel scale of 0.2 for nonconvex region. With the last classification algorithm of this chapter, CART algorithm, we examined the effect of the value of the minimum number of samples in the parent nodes on the task of periphery identification by posing two cases which included the minimum number of samples of 16 and one for parent nodes. Through these cases, it was shown that CART algorithm with minimum number of samples of one for parent nodes successfully separated feasible and infeasible points and gave a true nonparametric periphery almost equal to one obtained via k-NN classification algorithm with $k = 3$. In other words, likewise the k-NN classification algorithm, we achieved true, but highly zigzagged periphery when employing CART algorithm with minimum number of samples of one. To sum up, the task of the periphery identification of nonconvex region was successfully executed with five classification algorithms and we showed that PNN, k-NN

and CART algorithms may yield highly zigzagged peripheries, leading to completely true classification of feasible and infeasible points, depending on their hyperparameters.

Chapter 11 was dedicated to the topic of the extraction of the mathematical equation(s) of the boundary of a BW image relying on the technique of "data-based constraint identification". We firstly mentioned how to convert a BW image into a region formed by feasible and infeasible points. Then, we provided four BW images, which were named as "droplet", "V-shaped cup", "disjoint regions", and "human", respectively. Through processing these BW images, we obtained different types of regions, for instance, a contiguous feasible region with the "droplet" image, a disjoint infeasible region with the "V-shaped cup" image, a fragmented feasible region with the "disjoint regions" image, a disjoint feasible region with the "human" image. We used four different regions in four constraint-identification examples and employed a contiguous feasible region in one approximate identification example. For the approximate-identification example, we utilized the form-free constitutives. For the other (non-approximate) identification examples, we deployed single 3^{rd}-order polynomial constraint, single 7^{th}-order polynomial constraint, single neural constitutive build via NN with ReLU activation function, and single neural constitutive build via ELM with unorthodox activation function, respectively. With these examples, we proved that it is possible to successfully obtain the mathematical equation(s) of the boundary (or boundaries) of BW images.

Chapter 12 accommodated the chemical engineering applications of the method of "data-based constraint identification" including the identification of the feasible-operation region of a CSTR, the extraction of the mathematical equation of a phase-equilibrium envelope (with and without measurement errors), the identification of the feasible-operation region of Williams-Otto Plant (WOP). We showed that our algorithm successfully identifies the feasible operation region of CSTR with the utilization of constitutive constraints in Section 12.1, and successfully achieves the task of "plant-wide constraint identification" by the use of constitutive linear-interaction constraints together with the constitutive bound constraints in Section 12.4. For these two cases, we solved the mass-balance equations (equality constraints) via MATLAB's fsolve function, which enabled us to solve algebraic equations simultaneously, for each of the random sample points. Thus, we successfully determined the feasible-operation regions by previously

mentioned constitutives without violating equality constraints (mass balances) as well. By Section 12.2, using the Ethane-Benzene phase equilibrium data, we provided a method of generation of feasible and infeasible points from the experimental data, and showed a successful identification of the mathematical equation of this phase equilibrium envelope. Additionally, in Section 12.3, we proved that our algorithm can also obtain the mathematical equation of phase equilibrium envelope when the experimental data contains measurement error as nearly identical to the error-free one, when suitable constitutives (low order polynomial constitutives to filter out measurement errors) for such situations are utilized.

Overall, the major contribution of this thesis work of the "data-based constraint identification" is the development of a unique optimization formulation that embraces an overall objective function relying on the technique of "Constraint Aggregation" exclusively. In this thesis work, we primarily showed that this optimization formulation worked very well as intended for the identification of whatever the types of D-dimensional feasible or infeasible regions are, i.e., the contiguous feasible and infeasible regions, disjoint infeasible regions, and fragmented feasible regions.

13.2. Recommendations

In this thesis work, many constraint-identification examples were solved. However, we mostly exemplified the identification of two-dimensional feasible regions. This thesis work only accommodated two constraint-identification examples in n-D and the identification of three-dimensional feasible regions was not exemplified with the utilization of some types of constitutive constraints such as the form-free implicit neural constitutives. The technique of "data-based constraint identification" could have been applied to more diverse feasible regions in 3-D and n-D together with the utilization of all types of the constitutives developed in this work.

For the constraint-identification task, the procedure of an adaptive or stepwise sampling can be developed. For instance, our algorithm can begin with very few sample points, i.e., 50 sample points. After reaching the optimal values of the parameters of constitutives (optimal values of optimization decision variables) with 50 sample points, our

algorithm can repeat the optimization with more sample points, i.e., 100 sample points, by using the optimal values of optimization decision variables obtained with 50 sample points as initial guess. This topic can be further investigated and developed for a future work of "sample-adaptive data-based constraint identification".

Through this thesis work, we employed the numerically-stabilized, underestimator type Kreisselmeier-Steinhauser (KS) aggregation function. The utilization of the other aggregation functions such as Induced Exponential (IE), Induced Power (IP) aggregation functions can be studied for future constraint-identification works. However, we perceive and claim that the effects of above-mentioned alternative aggregations may not be observed at all.

The overall objective function introduced in Chapter 5 had three sub-objective functions and our algorithm could identify the feasible and infeasible regions mainly by means of the first and second sub-objective functions, i.e., J_1 and J_2, working together synergistically. Our algorithm could not detect the regions when one of them was utilized alone. It was clear why our algorithm failed to identify regions when J_2 was utilized alone; there is no optimizer that can handle the stagewise changing objective functions (stepwise decreases or increases in the integer-valued objective function values). However, it was not clear the failure of our algorithm when J_1 was utilized alone. This topic must be further investigated.

In Chapter 7, we mainly built certain types of form-free constitutives via design matrix approach and focused mostly on the utilization of these constitutives through the constraint-identification examples presented in Chapter 8. We could have utilized higher-order, for example, 10^{th}-order, polynomial constitutive constraints built via design matrix approach using the factorial-design method. In future studies, the building blocks of design matrix models can be created by utilizing, for instance, exponential functions, and thus, the form-free "exponential" constitutive constraints can be built for the identification of much more complex regions. Additionally, a single constitutive, whose building blocks were formed with the utilization of different functions together (e.g., single constitutive with 2^{nd} order polynomial terms together with exponential and/or logarithmic terms), can be created and utilized for the constraint-identification task in future works.

For all the constraint-identification examples of Chapter 7, our algorithm identified feasible regions by employing all the predefined constitutive constraints. For future works, the binary-decision variables can be included in our algorithm for the selection of constitutive constraints (or selection of significant terms in a constitutive constraint) by switching from the current "global NLP" to "global MINLP" optimization formulation. In this way, our algorithm can determine the most suitable combination of constitutive constraints within a predefined set of constitutives in accordance with the feasible region to be identified.

For the further development of the utilization of single constitutives via design matrix approach, our algorithm can begin with a low-order polynomial constitutive such as the 2^{nd}-order polynomial constitutive. If the optimization terminates with a non-zero objective value, our algorithm can repeat the procedure stagewise with a higher-order polynomial constitutive such as the 3^{rd}-order polynomial constitutive. In the second stage, optimal values of the parameters of the 2^{nd}-order polynomial constitutive from the previous stage can be used as initial guesses for the identical 2nd-order terms of the 3^{rd}-order polynomial constitutive. The parameter vector can then be augmented with zeros as initial guesses for the parameter locations specific to the remaining terms of the 3^{rd}-order polynomial constitutive. Such a stepwise increase in the order of polynomial constitutive can repeatedly be done until the optimization terminates with the objective function value of zero.

By taking the concept of "approximate constraint-identification" into consideration, an example on approximation of a convex circular feasible region via linear constitutives forming a convex octagonal feasible region was presented in Chapter 8. The deployment of our algorithm to reveal convex outer envelopes of nonconvex regions, just like convex hull algorithms such as Delaunay Triangulation, must be assessed in a future work.

We built the neural constitutive constraints by taking advantage of two ML tools (NN and ELM) that contain single hidden layers. For a future work of "data-based constraint identification", ML tools that consist of multiple hidden layers (possibly with fewer number of neurons in each layer) can be deployed for the creation of the neural constitutive constraints. In this way, our algorithm may gain much more flexibility to

identify much more complex feasible regions. Additionally, we were not concerned with finding the optimal topology of the ML tools (i.e., elimination of insignificant neurons) for the identification of the feasible regions. This topic can be considered in the scope of the future work of this thesis work.

We executed the task of the periphery identification by classification of feasible and infeasible regions (points) via some ML algorithm in Chapter 10. As a result of the periphery identification, one cannot obtain the mathematical equation(s) of actual inequality constraint(s), but the data on the periphery (contour data). By taking these data obtained through ML classification (e.g., PNN) and without using any feasible and infeasible points, our algorithm, redesigned with a new suitable objective function for the contour data, can extract the mathematical equation(s) of actual inequality constraint(s) very fast. This topic can also be studied in future works.

In this thesis work, we were only concerned with the identification of the inequality constraints of an optimization problem relying basically on feasible and infeasible points and the objective function of an optimization problem remained outside the scope of this work. The integration of our algorithm with a surrogate model, which can be replaced as implicit objective function of an optimization problem (e.g., Gaussian Process Regression), can be among the future works, e.g., a study on hybrid use of objective function identification via surrogate model generation and our constraint identification algorithm.

In Chapter 12, we exemplified a constraint-identification example with a feasible region under uncertainty (a region formed by the Ethane-Benzene data including measurement errors). By this example, we proved the success of our algorithm to filter out the noise from an experimental data when parsimonious constitutives was used, and thus, its capability to identify a feasible region nearly identical to true one even though uncertainty exhibited itself in the region-sampling phase. Considering its success of the identification of inequality constraints in the presence of uncertainty, one of the future investigation topics may be whether our algorithm can be used as an alternative approach to extract inequality constraints via constitutives in “chance-constrained optimization” problems.

Computational work to visualize the aggregated constraints were depended upon the zero-valued contour, generated via MATLAB's "contour" function throughout this thesis work. However, contour-generation algorithms are based on some sort of interpolation, and thus, for some cases, may not give an utterly exact visualization of the aggregated constraints, even though the feasible regions represented by aggregated constraints and optimal constitutives overlap each other in all examples of this thesis work. Thus, an alternative way to visualize the aggregated constraint or single polynomial constraints must be investigated to get rid of the dependency of our algorithm on contour generation. At this point, we should stress that our algorithmic steps do not use contour information at all; contour generation is used solely for final visualization purposes.

We had mentioned in Chapter 8 that we can compute the areas of closed sets of feasible or infeasible regions, whether they are contiguous or disjoint, particularly for 2-D problems. In future works, this bonus area information may be used to calculate the flexibility index of an inequality-constrained system based on a nominal operation point within the closed set of feasible region, which is still one of the hot topics in process systems engineering.

REFERENCES

Adi, V. S. K., Laxmidewi, R., and Chang, C.-T., 2016, "An Effective Computation Strategy for Assessing Operational Flexibility of High-Dimensional Systems with Complicated Feasible Regions", *Chemical Engineering Science*, Vol. 147, No. 1, pp. 137–149.

Ahmad, M. F., Isa, N. A. M., Lim, W. H., and Ang, K. M., 2022, "Differential Evolution: A Recent Review Based on State-of-the-Art Works", *Alexandria Engineering Journal*, Vol. 61, No. 5, pp. 3831–3872.

Al, R., Behera, C. R., Gernaey, K. V, and Sin, G., 2020, "Stochastic Simulation-Based Superstructure Optimization Framework for Process Synthesis and Design Under Uncertainty", *Computers & Chemical Engineering*, Vol. 143, No. 1, p. 107118.

Astorino, A., Fuduli, A., and Gaudioso, M., 2016, "Nonlinear Programming for Classification Problems in Machine Learning" *AIP Conference Proceedings*, Vol. 1776, No. 1, Pizzo Calabro, Italy.

Badejo, O., and Ierapetritou, M., 2022, "Integrating Tactical Planning, Operational Planning and Scheduling Using Data-Driven Feasibility Analysis", *Computers & Chemical Engineering*, Vol. 161, No. 1, p. 107759.

Banerjee, I., and Ierapetritou, M. G., 2002, "Design Optimization under Parameter Uncertainty for General Black-Box Models", *Industrial & Engineering Chemistry Research*, Vol. 41, No. 26, pp. 6687–6697.

Banerjee, I., Pal, S., and Maiti, S., 2010, "Computationally Efficient Black-Box Modeling for Feasibility Analysis", *Computers & Chemical Engineering*, Vol. 34, No. 9 ,pp. 1515–1521.

Basudhar, A., Dribusch, C., Lacaze, S., and Missoum, S., 2012, "Constrained Efficient Global Optimization with Support Vector Machines", *Structural and Multidisciplinary Optimization*, Vol. 46, No. 1, pp. 201–221.

Basudhar, A., and Missoum, S., 2010, "An Improved Adaptive Sampling Scheme for the Construction of Explicit Boundaries", *Structural and Multidisciplinary Optimization*, Vol. 42, No. 1, pp. 517–529.

Basudhar, A., Missoum, S., and Sanchez, A. H., 2008, "Limit State Function Identification Using Support Vector Machines for Discontinuous Responses and Disjoint Failure Domains", *Probabilistic Engineering Mechanics*, Vol. 23, No. 1, pp. 1–11.

Ben-Hur, A., and Weston, J., 2010, "A User's Guide to Support Vector Machines", *Methods in Molecular Biology*, Vol. 609, No. 1, pp. 223–239.

Benjelloun, M., Oliva, H. J. T., and Prevot, R., 2007, "Edge Closing of Synthetic and Real Images Using Polynomial Fitting", *Journal of Convergence Information Technol*ogy, Vol. 2, No. 4, pp. 8–19.

Benko, P., Kós, G., Várady, T., Andor, L., and Martin, R., 2002, "Constrained Fitting in Reverse Engineering", *Computer Aided Geometric Design*, Vol. 19, No. 3, pp. 173–205.

Bloss, K. F., Biegler, L. T., and Schiesser, W. E., 1999, "Dynamic Process Optimization through Adjoint Formulations and Constraint Aggregation", *Industrial & Engineering Chemistry Research*, Vol. 38, No. 2, pp. 421–432.

Boukouvala, F., Muzzio, F. J., and Ierapetritou, M. G., 2011, "Feasibility Analysis of Black-Box Processes Using an Adaptive Sampling Kriging Based Method", *Computer Aided Chemical Engineering*, Vol. 29, No. 1, pp. 432–436.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A., 1984, *Classification and Regression Trees*, Taylor & Francis, New York.

Chen, W., and Fuge, M., 2017, "Beyond the Known: Detecting Novel Feasible Domains over an Unbounded Design Space", *Journal of Mechanical Design*, Vol. 139, No. 11, p. 111405.

Cover, T., and Hart, P., 1967, "Nearest Neighbor Pattern Classification", *IEEE Transactions on Information Theory*, Vol. 13, No. 1, pp. 21–27.

Dabbene, F., Gay, P., and Polyak, B. T., 2003, "Recursive Algorithms for Inner Ellipsoidal Approximation of Convex Polytopes", *Automatica*, Vol. 39, No. 10, pp. 1773–1781.

Edgar, T. F., Himmelblau, D. M., and Lasdon, L. S., 2001, *Optimization of Chemical Processes*, McGraw-Hill, New York.

Fang, D., Zhang, T., and Wu, F., 2022, "An Active-Learning Probabilistic Neural Network for Feasibility Classification of Constrained Engineering Optimization Problems", *Engineering with Computers*, Vol. 38, No. 4, pp. 3237–3250.

Garud, S. S., Karimi, I. A., and Kraft, M., 2017, "Design of Computer Experiments: A Review", *Computers & Chemical Engineering*, Vol. 106, No. 1, pp. 71–95.

Ghobadi, K., and Mahmoudzadeh, H., 2021, "Inferring Linear Feasible Regions Using Inverse Optimization", *European Journal of Operational Research*, Vol. 290, No. 3, pp. 829–843.

Goyal, V., and Ierapetritou, M. G., 2002, "Determination of Operability Limits Using Simplicial Approximation", *AIChE Journal*, Vol. 48, No. 12, pp. 2902–2909.

Goyal, V., and Ierapetritou, M. G., 2003a, "Framework for Evaluating the Feasibility/Operability of Nonconvex Processes", *AIChE Journal*, Vol. 49, No. 5, pp. 1233–1240.

Goyal, V., and Ierapetritou, M. G., 2003b, "Integration of Data Analysis and Design Optimization for the Systematic Generation of Equipment Portfolio", *Industrial &*

Engineering Chemistry Research, Vol. 42, No. 21, pp. 5204–5214.

Han, F., Jiang, J., Ling, Q.-H., and Su, B.-Y., 2019, "A Survey on Metaheuristic Optimization for Random Single-Hidden Layer Feedforward Neural Network", *Neurocomputing*, Vol. 335, No. 1, pp. 261–273.

Hansen, N., Müller, S., and Koumoutsakos, P., 2003, "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)", *Evolutionary Computation*, Vol. 11, No.1, pp. 1–18.

Hu, B.-G., and Qu, H.-B., 2020, "Generalized Constraints as a New Mathematical Problem in Artificial Intelligence: A Review and Perspective", ArXiv:2011.06156.

Huang, C., 2020, "ReLU Networks Are Universal Approximators via Piecewise Linear or Constant Functions", *Neural Computation*, Vol. 32, No. 11, pp. 2249–2278.

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K., 2004, "Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks", *2004 IEEE International Joint Conference on Neural Networks*, Vol. 2, Budapest, Hungary, pp. 985–990.

Ierapetritou, M. G., 2001, "New Approach for Quantifying Process Feasibility: Convex and 1-D Quasi-Convex Regions", *AIChE Journal*, Vol. 47, No. 6, pp. 1407–1417.

Jeong, S.-H., Choi, D.-H., and Jeong, M., 2012, "Feasibility Classification of New Design Points Using Support Vector Machine Trained by Reduced Dataset", *International Journal of Precision Engineering and Manufacturing*, Vol. 13, No. 1, pp. 739–746.

Johnson, R. A., and Wichern, D. W., 2007, *Applied Multivariate Statistical Analysis*, Sixth Edition, Pearson Prentice Hall, New Jersey.

Kennedy, G. J., and Hicken, J. E., 2015, "Improved Constraint-Aggregation Methods", *Computer Methods in Applied Mechanics and Engineering*, Vol. 289, No. 1, pp. 332–354.

Khattree, R., and Naik, D. N., 2000, *Multivariate Data Reduction and Discrimination with SAS Software*, John Wiley & Sons, Inc., New York.

Knudde, N., Couckuyt, I., Shintani, K., and Dhaene, T., 2019, "Active Learning for Feasible Region Discovery", *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, Boca Raton, FL, USA, pp. 567–572.

Kovács, I., and Várady, T., 2020, "Constrained Fitting with Free-Form Curves and Surfaces", *Computer-Aided Design*, Vol. 122, No. 1, p. 102816.

Kovács, I., Várady, T., and Salvi, P., 2015, "Applying Geometric Constraints for Perfecting CAD Models in Reverse Engineering", *Graphical Models*, Vol. 82, No. 1 , pp. 44–57.

Kreisselmeier, G., and Steinhauser, R., 1980, "Application of Vector Performance: Optimization to a Robust Control Loop Design for a Fighter Aircraft", *International Journal of Control*, Vol. 37, No. 2, pp. 251-284.

Kucherenko, S., Giamalakis, D., Shah, N., and García-Muñoz, S., 2020, "Computationally Efficient Identification of Probabilistic Design Spaces through Application of Metamodeling and Adaptive Sampling", *Computers & Chemical Engineering*, Vol. 132, No. 1, p. 106608.

Kumar, M., Kolb, S., and Guns, T., 2022, "Learning Constraint Programming Models from Data Using Generate-And-Aggregate", *28th International Conference on Principles and Practice of Constraint Programming (CP 2022)*, edited by Solnon C., Haifa, Israel, pp. 32:1-32:16.

Kusumo, K. P., Gomoescu, L., Paulen, R., García Muñoz, S., Pantelides, C. C., Shah, N., and Chachuat, B., 2019, "Bayesian Approach to Probabilistic Design Space Characterization: A Nested Sampling Strategy", *Industrial & Engineering Chemistry Research*, Vol. 59, No. 6, pp. 2396–2408.

Lambe, A. B., Kennedy, G. J., and Martins, J. R. R. A., 2017, "An Evaluation of Constraint Aggregation Strategies for Wing Box Mass Minimization", *Structural and Multidisciplinary Optimization*, Vol. 55, No. 1, pp. 257–277.

Liu, D., Nosovskiy, G. V., and Sourina, O., 2008, "Effective Clustering and Boundary Detection Algorithm Based on Delaunay Triangulation", *Pattern Recognition Letters*, Vol. 29, No. 9, pp. 1261–1273.

Ludl, P. O., Heese, R., Höller, J., Asprion, N., and Bortz, M., 2022, "Using Machine Learning Models to Explore the Solution Space of Large Nonlinear Systems Underlying Flowsheet Simulations with Constraints", *Frontiers of Chemical Science and Engineering*, Vol. 16, No. 2, pp. 183–197.

Metta, N., Ramachandran, R., and Ierapetritou, M., 2020, "A Novel Adaptive Sampling Based Methodology for Feasible Region Identification of Compute Intensive Models Using Artificial Neural Network", *AIChE Journal*, Vol. 67, No. 2, p. e17095.

Montgomery, D. C., and Runger, G. C., 2018, *Applied Statistics and Probability for Engineers*, John Wiley & Sons, Inc., New York.

Morito, S., Koida, J., Iwama, T., Sato, M., and Tamura, Y., 1999, "Simulation-Based Constraint Generation with Applications to Optimization of Logistic System Design", *Proceedings of the 31st Conference on Winter Simulation: Simulation---a Bridge to the Future*, Vol. 1, Phoenix, AZ, USA, pp. 531–536.

Na, J., Lim, Y., and Han, C., 2017, "A Modified DIRECT Algorithm for Hidden Constraints in an LNG Process Optimization" *Energy*, Vol. 126, No. 1, pp. 488–500.

Nagar, D., Pannerselvam, K., and Ramu, P., 2022, "A Novel Data-Driven Visualization of n-Dimensional Feasible Region Using Interpretable Self-Organizing Maps (iSOM)", *Neural Networks*, Vol. 155, No. 1, pp. 398–412.

Patel, J., and Choi, S.-K., 2012, "Classification Approach for Reliability-Based Topology

Optimization Using Probabilistic Neural Networks", *Structural and Multidisciplinary Optimization*, Vol. 45, No. 1, pp. 529–543.

Price, K., Storn, R., and Lampinen, J., 2005, *Differential Evolution-A Practical Approach to Global Optimization*, Springer, Berlin, Heidelberg.

Qing, J., Knudde, N., Couckuyt, I., Dhaene, T., and Shintani, K., 2020, "Batch Bayesian Active Learning for Feasible Region Identification by Local Penalization", *2020 Winter Simulation Conference (WSC)*, Orlando, FL, USA, pp. 2779–2790.

Rasmussen, C., and Williams, C., 2006, *Gaussian Process for Machine Learning*, MIT Press, Cambridge.

Raspanti, C. G., Bandoni, J. A., and Biegler, L. T., 2000, "New Strategies for Flexibility Analysis and Design Under Uncertainty", *Computers & Chemical Engineering*, Vol. 24, No. 9, pp. 2193–2209.

Rijckaert, M. J., and Martens, X. M., 1974, "Analysis and Optimization of the Williams-Otto Process by Geometric Programming", *AIChE Journal*, Vol. 20, No. 4, pp. 742–750.

Samarasinghe, S., 2006, *Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition*, Taylor & Francis Group, New York.

Samatin Njikam, A. N., and Zhao, H., 2016, "A Novel Activation Function for Multilayer Feed-Forward Neural Networks", *Applied Intelligence*, Vol. 45, No. 1, pp. 75–82.

Shi, Y., Lu, Z., Zhou, J., and Zio, E., 2020, "A Novel Time-Dependent System Constraint Boundary Sampling Technique for Solving Time-Dependent Reliability-Based Design Optimization Problems", *Computer Methods in Applied Mechanics and Engineering*, Vol. 372, No. 1, p. 113342.

Shields, M. D., and Zhang, J., 2016, "The Generalization of Latin Hypercube Sampling", *Reliability Engineering & System Safety*, Vol. 148, No. 1, pp. 96–108.

Singh, P., Herten, J. van der, Deschrijver, D., Couckuyt, I., and Dhaene, T., 2017, "A Sequential Sampling Strategy for Adaptive Classification of Computationally Expensive Data", *Structural and Multidisciplinary Optimization*, Vol. 55, No. 1, pp. 1425–1438.

Sirdeshpande, A. R., Ierapetritou, M. G., Andrecovich, M. J., and Naumovitz, J. P., 2005, "Process Synthesis Optimization and Flexibility Evaluation of Air Separation Cycles", *AIChE Journal*, Vol. 51, No. 4, pp. 1190–1200.

Straus, J., Ouassou, J. A., Knudsen, B. R., and Anantharaman, R., 2021, "Constrained Adaptive Sampling for Domain Reduction in Surrogate Model Generation: Applications to Hydrogen Production", *AIChE Journal*, Vol. *67*, *No.* 11, p. e17357.

Wang, J., Smith, R., and Zhu, L., 2022, "An Adaptive Refined Grid Search Strategy for Assessing Operational Flexibility and Application on Refrigerant Selection", *AIChE Journal*, Vol. 68, No. 4, p. e17566.

Wasserman, P. D., 1993, *Advanced Methods in Neural Computing*, John Wiley & Sons, Inc., New York.

Williams, T. J., and Otto, R. E., 1960, "A Generalized Chemical Processing Model for the Investigation of Computer Control", *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, Vol. 79, No. 5, pp. 458–473.

Wu, G., 2015, "A Contour Fitting Algorithm for Rapid Prototyping Technology" *5th International Conference on Information Engineering for Mechanics and Materials*, Huhhot, Inner Mongolia, pp. 585–588.

Wu, Z., Chen, Z., Chen, G., Li, X., Jiang, C., Gan, X., Gao, L., and Wang, S., 2021, "A

Probability Feasible Region Enhanced Important Boundary Sampling Method for Reliability-Based Design Optimization", *Structural and Multidisciplinary Optimization*, Vol. 63, No. 1, pp. 341–355.

Zhang, L., Xu, Z.-Q. J., and Zhang, Y., 2022, "Data-Informed Deep Optimization", *Plos One*, Vol. 17, No. 6, p. e0270191.

Zhang, Q., Grossmann, I. E., Sundaramoorthy, A., and Pinto, J. M., 2016, "Data-Driven Construction of Convex Region Surrogate Models", *Optimization and Engineering*, Vol. 17, No. 1, pp. 289–332.

Zhao, F., and Chen, X., 2018, "Analytical and Triangular Solutions to Operational Flexibility Analysis Using Quantifier Elimination", *AIChE Journal*, Vol. 64, No. 11, pp. 3894-3911.

Zhao, F., Grossmann, I. E., García-Muñoz, S., and Stamatis, S. D., 2022, "Design Space Description through Adaptive Sampling and Symbolic Computation", *AIChE Journal*, Vol. 68, No. 5, p. e17604.

Zhao, F., Ochoa, M. P., Grossmann, I. E., García-Muñoz, S., and Stamatis, S. D., 2022, "Novel Formulations of Flexibility Index and Design Centering for Design Space Definition", *Computers & Chemical Engineering*, Vol. 166, No. 1, p. 107969.

Zhao, F., Zheng, C., Zhang, S., Zhu, L., and Chen, X., 2019, "Quantification of Process Flexibility via Space Projection", *AIChE Journal*, Vol. 65, No. 10, p. e16706.

Zheng, C., Zhao, F., Zhu, L., and Chen, X., 2020, "Operational Flexibility Analysis of High-Dimensional Systems via Cylindrical Algebraic Decomposition", *Industrial & Engineering Chemistry Research*, Vol. 59, No. 10, pp. 4670–4687.

APPENDIX

Appendix A.1. MATLAB Code for Example in Section 6.6

```
echo off; clc; clear all; close all; format short g; warning off;

% === CONFITLIB - Bounds+Linears+SphElps+Prbl


global NPts nDim nConLin nConSph nConPrbl
global term1 term2 term3 term4 term5 term6 term7 term8 term9 ABnds
global X Y nPtsX nPtsY Rho xLB xUB
global iplot remplot nMesh X1 X2

iplot=0; remplot=5000;

nDim = 2
nConBnds= 2*nDim
nConLin = 1
nConSph = 1
nConPrbl = 1

NPts = 5000;
Rho = 500;

xLB = [-1.0  -2.0]*1;
xUB = [+9.0  +7.0]*1;

term1=2*nDim*1;
term2=term1+nConLin*nDim;
term3=term2+nConLin;
term4=term3+nDim*nConSph;
term5=term4+nConSph;
term6=term5+nConSph;
term7=term6+nConPrbl;
term8=term7+nConPrbl*nDim;
term9=term8+nConPrbl;

ABnds = kron(eye(nDim),[1 -1]');


% --- Generate mesh for intermediate and final plotting
nMesh = 500;
[X1,X2] = meshgrid(linspace(xLB(1),xUB(1),nMesh), linspace(xLB(2),xUB(2),nMesh));
% -END- Generate mesh for intermediate and final plotting


% --- Sample the Region (Feasible & Infeasible)
X=[]; Y=[];
nStrata = ceil(repmat(NPts^(1/nDim),1,nDim)); [R] = LSS(nStrata); % Latinized Stratified Sampling

x = xLB + (xUB-xLB).*R;
g = constraints(x);
X = [x(all(((g <= 0) == 1), 2) == 1, :)]';
Y = [x(all(((g <= 0) == 1), 2) == 0, :)]';
```

```
clear g x R xm1 xm2
% % -END- Sample the Region (Feasible & Infeasible)
nPtsX = size(X,2)
nPtsY = size(Y,2)
NPts = nPtsX+nPtsY

% % --- Reduce number of trial points
pctl = 1;
nDist=round(prctile([1:min(length(X),length(Y))], pctl))
[D,Idx]=pdist2(X',Y','euclidean','Smallest',nDist);
IdxX=unique(Idx(:));
X=X(:,IdxX);
[D,Idx]=pdist2(Y',X','euclidean','Smallest',nDist);
IdxY=unique(Idx(:));
Y=Y(:,IdxY);
% -END- Reduce number of trial points
nPtsX = size(X,2)
nPtsY = size(Y,2)
NPts = nPtsX+nPtsY

Pars0 = randn(term9,1)' % 'rand' 'randn' 'rands' 'randnr' 'randnc' 'randsmall'
LB = -10*ones(length(Pars0),1)'; UB = +10*ones(length(Pars0),1)';
NP=40; IterMax = 150000; TolF=1.E-9; CR=0.6; F=0.8; Strategy=5; Refresh=100; Ifplot=0;
[ParsOpt ObjFctVal] = devecUADM(@ConFitObj,-inf,Pars0,LB,UB,NP,IterMax,TolF,F,CR,Strategy,Refresh,Ifplot)


ParsOpt'
ObjFctVal

iplot=-1;

FinalObjEvaluation = ConFitObj(ParsOpt)


% --- Bounds & Linears
bBnds(:,1) = ParsOpt(1:term1);
ALin = reshape(ParsOpt(1+term1:term2),nDim,nConLin)';
bLin(:,1) = reshape(ParsOpt(1+term2:term3),nConLin,1);
A = [ABnds; ALin]
b = [bBnds; bLin]
% -END- Bounds & Linears

% --- Perfect or Elongated Sphere without Rotation
c(:,:) = reshape(ParsOpt(1+term3:term4),nDim,nConSph)
e(:,1) = reshape(ParsOpt(1+term4:term5),nConSph,1)
r(:,1) = reshape(ParsOpt(1+term5:term6),nConSph,1)
e_r = e./r
% -END- Perfect or Elongated Sphere without Rotation

%%% Parabola
c_prbl(:,1) = reshape(ParsOpt(1+term6:term7),nConPrbl,1)
e_prbl(:,:) = reshape(ParsOpt(1+term7:term8),nDim,nConPrbl)
r_prbl(:,1) = reshape(ParsOpt(1+term8:term9),nConPrbl,1)
%%% -END- Parabola


% --- Constraint Aggregation
vecMesh=[reshape(X1,nMesh^nDim,1) reshape(X2,nMesh^nDim,1)]';
termBLX = A*vecMesh-b; % All must be <= 0 &  Dimension must be (2*nDimn+ConLin) x nDat

for i=1:nConSph % e1*(x1-c1)^2 + e2*(x2-c2)^2 < r
   termSphX(i,:) = sum( e(i)*[vecMesh-c(:,i)].^2, 1) - r(i); % SUM over nDim & Dimension must be nConSph x nDat
```

```
end

%%% Parabola
for i = 1:nConPrbl
   termPrblX(i,:) =  sum( e_prbl(1,i)*[vecMesh(1,:)-c_prbl(i)].^2 -e_prbl(2,i)*vecMesh(2,:), 1)+ r_prbl(i);
end
%%% -END- Parabola



termALLX=[termBLX;termSphX;termPrblX];
maxtermBLSphElpsX=max(termALLX,[],1);
CAMesh = maxtermBLSphElpsX+(1/Rho)*log( sum( exp(Rho*(termALLX-maxtermBLSphElpsX)), 1 ) ); % All must be <= 0

Z=(reshape(CAMesh,nMesh,nMesh));


fig1=figure(1);
plot(X(1,:),X(2,:),'bo','MarkerSize',2,'MarkerFaceColor','b'); hold on; plot(Y(1,:),Y(2,:),'ro','MarkerSize',2);
xlabel('z_1','FontName','.','FontSize',11);ylabel('z_2','FontName','.','FontSize',11)
axis equal

set(gca,'Xlim',[xLB(1) xUB(1)],'Ylim',[xLB(2) xUB(2)]);

% % %
hTxt = text(7.1,-1.1,'\rmg_{\fontsize{7.5}1}(\rmθ_{\rm\fontsize{7.5}1}\rm)','fontsize',10.5,'color','k');%7.1, -1.1  %7.1,4.1
hTxt(2)=text(7.11,-1.03,'\bf\wedge','VerticalAlignment','bottom','FontSize',7.0);%7.09,-1.03 %7.09,4.17

hTxt = text(-0.15,6.0,'\rmg_{\fontsize{7.5}2}(\rmθ_{\rm\fontsize{7.5}2}\rm)','fontsize',10.5,'color','k'); %-0.15,6.0 %-
0.15,0.9
hTxt(2)=text(-0.16,6.07,'\bf\wedge','VerticalAlignment','bottom','FontSize',7.0);%-0.16,6.07 %-0.16,0.97
%
hTxt = text(2.31,5.55,'\rmg_{\fontsize{7.5}3}(\rmθ_{\rm\fontsize{7.5}3}\rm)','fontsize',10.5,'color','k');
hTxt(2)=text(2.30,5.62,'\bf\wedge','VerticalAlignment','bottom','FontSize',7.0);
%
hTxt = text(4.9,-0.6,'\rmg_{\fontsize{7.5}4}(\rmθ_{\rm\fontsize{7.5}4}\rm)','fontsize',10.5,'color','k');
hTxt(2)=text(4.89,-0.53,'\bf\wedge','VerticalAlignment','bottom','FontSize',7.0);

hTxt = text(7.6,2.8,'\rmg_{\fontsize{7.5}5}(\bfθ_{\rm\fontsize{7.5}5}\rm)','fontsize',10.5,'color','k');
hTxt(2)=text(7.61,2.87,'\bf\wedge','VerticalAlignment','bottom','FontSize',7.0);

hTxt = text(2.5,1.8,'\rmg_{\fontsize{7.5}6}(\bfθ_{\rm\fontsize{7.5}6}\rm)','fontsize',10.5,'color','k');
hTxt(2)=text(2.49,1.87,'\bf\wedge','VerticalAlignment','bottom','FontSize',7.0);

hTxt = text(4.55,6,'\rmg_{\fontsize{7.5}7}(\bfθ_{\rm\fontsize{7.5}7}\rm)','fontsize',10.5,'color','k');
hTxt(2)=text(4.54,6.07,'\bf\wedge','VerticalAlignment','bottom','FontSize',7.0);

hTxt = text(5.1,1.3,'\rmℂ\rm(\bfθ\rm)','fontsize',10.5,'color',[0 0.5 0]);

% color = {'r','b','y','g','m','k','c'};

for k = 1 : nConBnds + nConLin + nConSph+nConPrbl

   if k <= nConBnds
      if k < 3
        fcts{k} = @(x1,x2)  ABnds(k,1)*x1 + 0*x2-bBnds(k);
      else
        fcts{k} = @(x1,x2)  ABnds(k,2)*x2+ 0*x1 -bBnds(k);
      end
   elseif k > nConBnds & k <= nConBnds + nConLin
     fcts{k} = @(x1,x2)  ALin(k-nConBnds,1)*x1 + ALin(k-nConBnds,2)*x2 - bLin(k-nConBnds);
```

```
   elseif k > nConBnds + nConLin & k <= nConBnds + nConLin + nConSph
      fcts{k} = @(x1,x2)  e(k-nConBnds-nConLin)*( x1-c(1,k-nConBnds-nConLin))^2 +....
                  e(k-nConBnds-nConLin)*( x2-c(2,k-nConBnds-nConLin))^2 - r(k-nConBnds-nConLin);
   else
      fcts{k} = @(x1,x2)  e_prbl(1,k-nConBnds-nConLin-nConSph)*( x1-c_prbl(k-nConBnds-nConLin-nConSph))^2 -....
                  e_prbl(2,k-nConBnds-nConLin-nConSph)*x2 + r_prbl(k-nConBnds-nConLin-nConSph);
   end



fimplicit(fcts{k}, [xLB(1) xUB(1) xLB(2) xUB(2)],'MeshDensity',150,'LineWidth',2);
xlim=[xLB(1) xUB(1)]; ylim=[xLB(2) xUB(2)];
hold on;
end

v=[0 0];
[C,h] = contour(X1,X2,Z,v,'g-','LineWidth',2.3); grid off;
set(h,'ShowText','off');
tightfigUA();
% %-END- Constraint Aggregation


%  %=====
function Obj = ConFitObj(Pars)

global NPts nDim nConLin nConSph nConPrbl
global term1 term2 term3 term4 term5 term6 term7 term8 term9 ABnds
global X Y nPtsX nPtsY Rho xLB xUB
global iplot remplot nMesh X1 X2

% --- Bounds & Linears
bBnds(:,1) = Pars(1:term1);
ALin = reshape(Pars(1+term1:term2),nDim,nConLin)';
bLin(:,1) = reshape(Pars(1+term2:term3),nConLin,1);
A = [ABnds; ALin];
b = [bBnds; bLin];
% -END- Bounds & Linears

% --- Perfect or Elongated Sphere without Rotation
c(:,:) = reshape(Pars(1+term3:term4),nDim,nConSph);
e(:,1) = reshape(Pars(1+term4:term5),nConSph,1);
r(:,1) = reshape(Pars(1+term5:term6),nConSph,1);
% -END- Perfect or Elongated Sphere without Rotation

%%% Parabola
c_prbl(:,1) = reshape(Pars(1+term6:term7),nConPrbl,1);
e_prbl(:,:) = reshape(Pars(1+term7:term8),nDim,nConPrbl);
r_prbl(:,1) = reshape(Pars(1+term8:term9),nConPrbl,1);
%%% -END- Parabola


% clear termBLX termBLY termSphX termSphY
% --- Constraint Aggregation
termBLX = A*X-b; % All must be <= 0 &  Dimension must be (2*nDimn+ConLin) x nDat
termBLY = A*Y-b; % All must be <= 0 &  Dimension must be (2*nDimn+ConLin) x nDat

for i=1:nConSph % e1*(x1-c1)^2 + e2*(x2-c2)^2 < r
   termSphX(i,:) = sum( e(i)*[X-c(:,i)].^2, 1) - r(i); % SUM over nDim & Dimension must be nConSph x nDat
   termSphY(i,:) = sum( e(i)*[Y-c(:,i)].^2, 1) - r(i); % SUM over nDim & Dimension must be nConSph x nDat
end

%%% Parabola
```

```
for i = 1:nConPrbl
   termPrblX =  sum( e_prbl(1,i)*[X(1,:)-c_prbl(i)].^2 -e_prbl(2,i)*X(2,:), 1)+ r_prbl(i);
   termPrblY =  sum( e_prbl(1,i)*[Y(1,:)-c_prbl(i)].^2 -e_prbl(2,i)*Y(2,:), 1)+ r_prbl(i);
end
%%% -END-Parabola



termALLX=[termBLX;termSphX;termPrblX];
termALLY=[termBLY;termSphY;termPrblY];

maxtermBLSphElpsX=max(termALLX,[],1);
maxtermBLSphElpsY=max(termALLY,[],1);
% CAX & CAY : SUM over nConSph & final imension must be 1 x nDat
CAX = maxtermBLSphElpsX+(1/Rho)*log( sum( exp(Rho*(termALLX-maxtermBLSphElpsX)), 1 ) ); % All must be <= 0
CAY = maxtermBLSphElpsY+(1/Rho)*log( sum( exp(Rho*(termALLY-maxtermBLSphElpsY)), 1 ) ); % All must be >= 0
% -END- Constraint Aggregation


Obj = 1e+0*(sum(max(0,CAX)) + sum(max(0,-CAY))) + ...
     1e+0*(sum(CAX > 0) + sum(CAY <= 0)) + ...
     1e-7*norm(Pars,1);


% %--- Intermediate Plot
if (rem(iplot,remplot) == 0) | iplot == -1

if iplot == -1
   TERM1=(sum(max(0,CAX)) + sum(max(0,-CAY)))
   TERM2=(sum(CAX > 0) + sum(CAY <= 0))
   TERM3=1.e-7*norm(Pars,1)
   MINMAX_CA=[minmax(CAX)     minmax(CAY)]
   COUNTS_CA=[sum(CAX > 0)   sum(CAY <= 0)]
end

end
iplot=iplot+1;

end


%  =====
function g = constraints(x)
g(:,1) = (+x(:,1)-7);
g(:,2) = (-x(:,1)+1);
g(:,3) = (+x(:,2)-5);
g(:,4) = (-x(:,2));
g(:,5) = (+x(:,1)+x(:,2)-10);
g(:,6) = (-(x(:,1)-2.0).^2  - (x(:,2)-1.0).^2 + 4.0 );
g(:,7) = (-10*(x(:,1)-5).^2 + x(:,2) - 2);
end
```

Appendix A.2. MATLAB Code for Example in Section 8.2

```
cho off; clc; clear all; close all; format short g; warning off;
% === CONFITLIB

global Rho NPts nPtsX nPtsY nPars nConModel nConBnds nDim iplot remplot
global X Y DX  DY X1 X2 DMesh nMesh
```

```
iplot=0; remplot=5000;

nDim = 2
NPts = 3500;
Rho = 500;

% Model='linear' %  'linear' / 'interaction' / 'quadratic' / 'purequadratic'
ConModel = repelem([{'linear'},{'interaction'},{'purequadratic'},{'quadratic'}],...
                [1 0 2 0])'

nConBnds  = 2*nDim * 1
nConModel = length(ConModel)

xLB = [-1.0  -2.0]*1;
xUB = [+9.0  +7.0]*1;

% --- Generate mesh for intermediate and final plotting
nMesh = 250;
[X1,X2] = meshgrid(linspace(xLB(1),xUB(1),nMesh), linspace(xLB(2),xUB(2),nMesh));
vecMesh=[reshape(X1,nMesh^nDim,1) reshape(X2,nMesh^nDim,1)]';
DMesh=[];

if nConBnds ~= 0
dim1Mesh=ones(1,nMesh^nDim);
DMesh=[dim1Mesh; vecMesh(1,:); dim1Mesh; -vecMesh(1,:); dim1Mesh; vecMesh(2,:); dim1Mesh; -vecMesh(2,:)];
end

for k=1:nConModel
DXk = x2fx(vecMesh',string(ConModel(k)))';
DMesh=[DMesh; DXk];
clear DXk DYk
end
% -END- Generate mesh for intermediate and final plotting


% --- Sample the Region (Feasible & Infeasible)
nStrata = ceil([(NPts)^(1/nDim),(NPts)^(1/nDim)]);
[R] = LSS([nStrata]); % Latinized Stratified Sampling

x = xLB + (xUB-xLB).*R;
g = constraints(x);
X = [x(all(((g <= 0) == 1), 2) == 1, :)]';
Y = [x(all(((g <= 0) == 1), 2) == 0, :)]';

clear g x R xm1 xm2
% -END- Sample the Region (Feasible & Infeasible)

nPtsX = size(X,2)
nPtsY = size(Y,2)
NPts = nPtsX+nPtsY

RATnPtsX0=nPtsX/NPts
RATnPtsY0=nPtsY/NPts


% --- Reduce number of trial points
pctl = 1;
nDist=round(prctile([1:min(length(X),length(Y))], pctl));
[D,Idx]= pdist2(X',Y','euclidean','Smallest',nDist);
IdxX = unique(Idx(:));
X = X(:,IdxX);
```

```
[D,Idy] = pdist2(Y',X','euclidean','Smallest',nDist);
IdxY = unique(Idy(:));
Y = Y(:,IdxY);
% -END- Reduce number of trial points

nPtsX = size(X,2)
nPtsY = size(Y,2)
NPts = nPtsX+nPtsY

RATnPtsX=nPtsX/NPts
RATnPtsY=nPtsY/NPts


% --- Construct Design Matrices & Augment with External Design Matrices
if nConBnds ~= 0
dim1X=ones(1,nPtsX); dim1Y=ones(1,nPtsY);
DX = [dim1X; X(1,:); dim1X; -X(1,:); dim1X; X(2,:); dim1X; -X(2,:)];
DY = [dim1Y; Y(1,:); dim1Y; -Y(1,:); dim1Y; Y(2,:); dim1Y; -Y(2,:)];
end


nPars=ones(1,nConBnds);
for k=1:nConModel
  DXk = x2fx(X',string(ConModel(k)))';
  DYk = x2fx(Y',string(ConModel(k)))';
  % DX=normalize(DX,'range',[-1 1]);
  % DY=normalize(DY,'range',[-1 1]);
  nPars(1,length(nPars)+1) = size(DXk,1);
  DX = [DX;DXk];
  DY = [DY;DYk];
clear DXk DYk
end
% -END- Construct Design Matrices & Augment with External Design Matrices


% %---
Pars0 = randn(sum(nPars),1)'


LB = -10*ones(length(Pars0),1)'; UB = +10*ones(length(Pars0),1)';
NP=80; IterMax=250000; TolF=1.E-9; CR=0.5; F=0.5; Strategy=5; Refresh=250; Ifplot=0;
[ParsOpt ObjFctVal] = devecUA(@ConFitObj,-inf,Pars0,LB,UB,NP,IterMax,TolF,F,CR,Strategy,Refresh,Ifplot)


ParsOpt=ParsOpt(:)';

iplot=-1;
FinalObjEvaluation = ConFitObj(ParsOpt)



for k = 1 : nConBnds+nConModel
   cidx=1+sum(nPars(1:k-1)):sum(nPars(1:k));
   if k <= nConBnds
     ParsMat(k,cidx+(k-1):cidx+k) = [ParsOpt(k) 1];
   else
     ParsMat(k,cidx+nConBnds) = ParsOpt(cidx);
   end
end

ParsMat = sparse(ParsMat)
```

```
termConModelX = ParsMat*DMesh; % Dimension must be nConBnds+nConModel x nDat

maxtermConModelX=max(termConModelX,[],1);
CAMesh = maxtermConModelX+log( sum( exp(Rho*(termConModelX-maxtermConModelX)), 1 ) )/Rho; % All must be <=
0
Z=(reshape(CAMesh,nMesh,nMesh));
% -END- Constraint Aggregation



% %--- Plot Region Identified

fig1=figure(1);

plot(X(1,:),X(2,:),'bo','MarkerSize',2,'MarkerFaceColor','b'); hold on; plot(Y(1,:),Y(2,:),'ro','MarkerSize',2);
xlabel('z_1','FontName','.','FontSize',11);ylabel('z_2','FontName','.','FontSize',11)
axis equal

set(gca,'Xlim',[xLB(1) xUB(1)],'Ylim',[xLB(2) xUB(2)]);


Col = {[0.6350 0.0780 0.1840],[0.3010 0.7450 0.9330],[0.4940 0.1840 0.5560],...
     [0.9290 0.6940 0.1250],[0.8500 0.3250 0.0980],[0 0.4470 0.7410],...
     [0.4660 0.6740 0.1880]};

for k = 1 : nConBnds+nConModel
if k <= nConBnds
  if k == 1
    fcts{k} = @(x1,x2) ParsOpt(k) + x1;
  elseif k == 2
    fcts{k} = @(x1,x2) ParsOpt(k) - x1;
  elseif k == 3
    fcts{k} = @(x1,x2) ParsOpt(k) + x2;
  elseif k == 4
    fcts{k} = @(x1,x2) ParsOpt(k) + x2;
  end
else
cidx = 1+sum(nPars(1:k-1)):sum(nPars(1:k));
p = ParsOpt(cidx);
if strcmp(string(ConModel(k-nConBnds)),'linear')
  fcts{k} = @(x1,x2) p(1) + p(2)*x1 + p(3)*x2;
elseif strcmp(string(ConModel(k-nConBnds)),'interaction')
  fcts{k} = @(x1,x2) p(1) + p(2)*x1 + p(3)*x2 + p(4)*x1*x2;
elseif strcmp(string(ConModel(k-nConBnds)),'purequadratic')
  fcts{k} = @(x1,x2) p(1) + p(2)*x1 + p(3)*x2 + p(4)*x1^2 + p(5)*x2^2;
elseif strcmp(string(ConModel(k-nConBnds)),'quadratic')
  fcts{k} = @(x1,x2) p(1) + p(2)*x1 + p(3)*x2 + p(4)*x1*x2 + p(5)*x1^2 + p(6)*x2^2;
end
end
fimplicit(fcts{k}, [xLB(1) xUB(1) xLB(2) xUB(2)],'MeshDensity',150,'LineWidth',2,'Color',Col{k});
xlim=[xLB(1) xUB(1)]; ylim=[xLB(2) xUB(2)];
hold on;
end

hTxt = text(7.1,-
1.1,'\rmg_{\fontsize{7.5}1}(\bfθ_{\rm\fontsize{7.5}1}^{\rm\fontsize{7.5}UB}\rm)','fontsize',10.5,'color','k');
hTxt(2)=text(7.09,-1.03,'\bf\wedge','VerticalAlignment','bottom','FontSize',7.0);

hTxt = text(-
0.25,6.0,'\rmg_{\fontsize{7.5}3}(\bfθ_{\rm\fontsize{7.5}1}^{\rm\fontsize{7.5}LB}\rm)','fontsize',10.5,'color','k');
hTxt(2)=text(-0.26,6.07,'\bf\wedge','VerticalAlignment','bottom','FontSize',7.0);
```

```
hTxt =
text(2.31,5.63,'\rmg_{\fontsize{7.5}2}(\bfθ_{\rm\fontsize{7.5}2}^{\rm\fontsize{7.5}UB}\rm)','fontsize',10.5,'color','k');
hTxt(2)=text(2.30,5.70,'\bf\wedge','VerticalAlignment','bottom','FontSize',7.0);
%
hTxt = text(4.9,-
0.6,'\rmg_{\fontsize{7.5}4}(\bfθ_{\rm\fontsize{7.5}2}^{\rm\fontsize{7.5}LB}\rm)','fontsize',10.5,'color','k');
hTxt(2)=text(4.89,-0.53,'\bf\wedge','VerticalAlignment','bottom','FontSize',7.0);

hTxt = text(7.6,2.8,'\rmg_{\fontsize{7.5}5}(\bfθ^{\rm\fontsize{7.5}L}\rm)','fontsize',10.5,'color','k');
hTxt(2)=text(7.59,2.87,'\bf\wedge','VerticalAlignment','bottom','FontSize',7.0);

hTxt =
text(2.2,1.7,'\rmg_{\fontsize{7.5}6}(\bfθ_{\rm\fontsize{7.5}1}^{\rm\fontsize{7.5}PQ}\rm)','fontsize',10.5,'color','k');
hTxt(2)=text(2.19,1.77,'\bf\wedge','VerticalAlignment','bottom','FontSize',7.0);

hTxt =
text(4.45,6.2,'\rmg_{\fontsize{6.8}7}(\bfθ_{\rm\fontsize{6.8}2}^{\rm\fontsize{6.8}PQ}\rm)','fontsize',9.7,'color','k');
hTxt(2)=text(4.44,6.27,'\bf\wedge','VerticalAlignment','bottom','FontSize',6.4);

hTxt = text(5.1,1.3,'\rmℂ\rm(\bfP\rm)','fontsize',10.5,'color',[0 0.5 0]);

v=[0 0];
[C,h] = contour(X1,X2,Z,v,'g-','LineWidth',2.3); grid off;
set(h,'ShowText','off');

tightfigUA();
% %---END Plot Region Identified

% % --- Compute perimeter & Area
[s] = plotcontour(C);
[Perimeter_CA ,Area_CA] = interpclosed(s.xdata,s.ydata,'pchip') % only for contigious contour
%%--- END Compute perimeter & Area

%  =====
function Obj = ConFitObj(Pars)
% function [Obj ObjGrad] = ConFitObj(Pars)

global Rho NPts nPtsX nPtsY nPars nConModel nConBnds nDim iplot remplot
global X Y DX DY  X1 X2 DMesh nMesh


for k = 1 : nConBnds+nConModel
   cidx=1+sum(nPars(1:k-1)):sum(nPars(1:k));
   if k <= nConBnds
     p(k,cidx+(k-1):cidx+k) = [Pars(k) 1];
   else
     p(k,cidx+nConBnds) = Pars(cidx);
   end
end

p=sparse(p); % This is the fastest approach!
termConModelX = p*DX; % nConBnds+nConModel x nDat
termConModelY = p*DY; % nConBnds+nConModel x nDat

% --- Constraint Aggregation
maxtermConModelX = max(termConModelX,[],1);
maxtermConModelY = max(termConModelY,[],1);

% CAX & CAY : SUM over nConSph & final imension must be 1 x nDat
% --- Classic KS Aggregation
CAX = maxtermConModelX+log(sum(exp(Rho*(termConModelX-maxtermConModelX)),1))/Rho; % All must be <= 0
CAY = maxtermConModelY+log(sum(exp(Rho*(termConModelY-maxtermConModelY)),1))/Rho; % All must be >= 0
```

```
% -END- Classic KS Aggregation

% -END- Constraint Aggregation


Obj = 1e+0*(sum(max(0,CAX)) + sum(max(0,-CAY))) + ...
    1e+0*(sum(CAX > 0) + sum(CAY <= 0)) + ...
    1e-6*norm(Pars,1);

if iplot == -1
  TERM1 = (sum(max(0,CAX)) + sum(max(0,-CAY)))
  TERM2 = (sum(CAX > 0) + sum(CAY <= 0))
  TERM3 = 1.e-6*norm(Pars,1)
  TERM4 = norm(Pars,1)
  MINMAX_CA = [minmax(CAX) minmax(CAY)]
  COUNTS_CA = [sum(CAX > 0) sum(CAY <= 0)]
end
end


%  =====
function g = constraints(x)
g(:,1) = (+x(:,1)-7);
g(:,2) = (-x(:,1)+1);
g(:,3) = (+x(:,2)-5);
g(:,4) = (-x(:,2));
g(:,5) = (+x(:,1)+x(:,2)-10);
g(:,6) = (-(x(:,1)-2.0).^2-(x(:,2)-1.0).^2 + 4.0 );
g(:,7) = (-10*(x(:,1)-5).^2 + x(:,2)-2);
end
```

Appendix A.3. MATLAB Code for Example in Section 9.4

```
echo off; clc; clear all; close all; format short g; warning off;

global NetInput nNetOutput term1 term2 WI
global Rho iplot remplot NPts nPtsX nPtsY nMesh
global X Y X1 X2  C0

iplot=0; remplot=5000;

NPts = 2000;
nCon = 5;
nDim = 2;
Rho  = 500;

xLB = [-0.0  -1.0]*1;
xUB = [+8.0  +6.0]*1;


%--- NNet Initialization
nNetInput  = nDim; % no. of inputs
nNetOutput = 5; % no. of outputs
nNetHLN    = 10; % no. of neorons
term1=nNetOutput*(nNetHLN+1); term2=nNetHLN+1; % with Output Biasses
% term1=nNetOutput*(nNetHLN); term2=nNetHLN; % without Output Biasses
```

```
% --- Generate mesh for intermediate and final plotting
nMesh = 1500;
[X1,X2] = meshgrid(linspace(xLB(1),xUB(1),nMesh), linspace(xLB(2),xUB(2),nMesh));
NetInput=[reshape(X1,nMesh^nDim,1) reshape(X2,nMesh^nDim,1)]';


WI = (rand(nNetHLN,nNetInput+1)-0.5)*1
WO = (rand(nNetOutput,nNetHLN+1)-0.5)*1 % With Output Biasses
% WO = (rand(nNetOutput,nNetHLN)-0.5)*1 % Without Output Biasses

NumberTotalWeightsBiases=numel(WI)+numel(WO)

Pars0 = {WO};
Pars0 = [Pars0{1}(:)]'

NumberTotalOptimizationParameters=length(Pars0)
%-END- NNet Initialization

% --- Sample the Region (Feasible & Infeasible)
nStrata = ceil([(NPts)^(1/nDim),(NPts)^(1/nDim)]);
[R] = LSS([nStrata]); % Latinized Stratified Sampling


x = xLB + (xUB-xLB).*R;
g = constraints(x);

X = [x(all(((g <= 0) == 1), 2) == 1, :)]';
Y = [x(all(((g <= 0) == 1), 2) == 0, :)]';

nPtsX = size(X,2)
nPtsY = size(Y,2)
NPts = nPtsX+nPtsY

clear g x R xm1 xm2
% -END- Sample the Region (Feasible & Infeasible)

% --- Reduce number of trial points
pctl = 1;
nDistX=round(prctile([1:length(X)], pctl));
[D,Idx]=pdist2(X',Y','euclidean','Smallest',nDistX);
IdxX=unique(Idx(:));
X=X(:,IdxX);
pctl = 0.5;
nDistY=round(prctile([1:length(Y)], pctl));
[D,Idy]=pdist2(Y',X','euclidean','Smallest',nDistY);
IdxY=unique(Idy(:));
Y1=Y(:,IdxY);
sY=randperm(size(Y,2),round(size(Y,2)/6));
Y2=Y(:,sY);
Y=[Y1 Y2];
% -END- Reduce number of trial points

nPtsX = size(X,2)
nPtsY = size(Y,2)
NPts = nPtsX+nPtsY


% &--- Differential Evolution
LB = -1*ones(length(Pars0),1)'; UB = +1*ones(length(Pars0),1)';
NP=40; IterMax=4.068e+4; TolF=1.E-9; CR=0.6; F=0.8; Strategy=5; Refresh=100; Ifplot=0;
[ParsOpt ObjFctVal] = devecUADM(@ObjMatNN,-inf,Pars0,LB',UB',NP,IterMax,TolF,F,CR,Strategy,Refresh,Ifplot)
options = optimset('Display','none','MaxFunEvals',2000,'MaxIter',250,...
```

```
'TolFun',1.E-10,'TolX',1.E-10);

OptimalParameters=ParsOpt'
iplot=0;
OptimalObj=ObjMatNN(ParsOpt)
% &---END Differential Evolution


%--- NNet
[NetOutMesh] = MatNNet(ParsOpt,NetInput);
%-END- NNet

%--- Constraint Aggregation
[CAMesh]=ConAgg(nNetOutput,NetOutMesh,Rho);
%-END- Constraint Aggregation
ZMesh=(reshape(CAMesh,nMesh,nMesh));
COUNT_Mesh = sum(CAMesh >= 0)


%--- Compute Final Objective Value
%--- NNet
[NetOutX] = MatNNet(ParsOpt,X);
[NetOutY] = MatNNet(ParsOpt,Y);
%-END- NNet

%--- Constraint Aggregation
[CAX]=ConAgg(nNetOutput,NetOutX,Rho);
[CAY]=ConAgg(nNetOutput,NetOutY,Rho);
%-END- Constraint Aggregation
COUNT_X = sum(CAX > 0)
COUNT_Y = sum(CAY <= 0)
%-END- Compute Final Objective Value


%%--- Depict neural implicit constitutives and aggregated constraint
fig1=figure(1);
subtightplot(1,2,1,[-0.5,+0.01],[-0.1,-0.27],[-0.09,-0.29])
plot(X(1,:),X(2,:),'bo','MarkerSize',3,'MarkerFaceColor','b'); hold on; plot(Y(1,:),Y(2,:),'ro','MarkerSize',3);
xlabel('z_1','FontName','.','FontSize',13.5);ylabel('z_2','FontName','.','FontSize',13.5)
title('a) Region Identified by Aggregation of Five ELM Outputs','FontSize',11)
axis equal
set(gca,'Xlim',[xLB(1) xUB(1)],'Ylim',[xLB(2) xUB(2)]);

hTxt = text(+3.4,2.5,'\rmℂ\rm(\bfθ\rm)','fontsize',15,'color',[0 0.5 0]);

v=[-0 +0];
[C,h] = contour(X1,X2,ZMesh,v,'g-','LineWidth',3); grid off;
set(h,'ShowText','off');

% --- Contour Plots of Individual Network Outputs
clr = {'b','r','c','y','m','k',[.5 .6 .7],[.8 .2 .6]}; % Cell array of colors.
subtightplot(1,2,2,[-0.5,+0.01],[-0.1,-0.27],[0.01,-0.39])

legend_name = {'$\mathrm{\hat{g}_{1}(\mbox{\boldmath $\theta$})}$',...
'$\mathrm{\hat{g}_{2}(\mbox{\boldmath $\theta$})}$',...
'$\mathrm{\hat{g}_{3}(\mbox{\boldmath $\theta$})}$',...
'$\mathrm{\hat{g}_{4}(\mbox{\boldmath $\theta$})}$',...
'$\mathrm{\hat{g}_{5}(\mbox{\boldmath $\theta$})}$'}

for k=1:nNetOutput
gNetOutMesh=(reshape(NetOutMesh(k,:),nMesh,nMesh));
[C,h] = contour(X1,X2,gNetOutMesh,[0 0],'-','LineWidth',3); grid on;
```

```
h.LineColor=clr{k};
legendInfo{k}=[legend_name{k}];
hold on;
end

legendInfo{k+1}='$\mathrm{C(\mbox{\boldmath $\theta$})}$';
[C,h] = contour(X1,X2,ZMesh,[0 0],'k--','LineWidth',3); grid on;
h.LineColor='k';
legend(legendInfo,'Location','Best','Interpreter','latex','FontSize',15)
title('b) Five Individual Neural Inequalities','FontSize',11.5)
xlabel('z_1','FontName','.','FontSize',13.5);ylabel('z_2','FontName','.','FontSize',13.5)
axis equal
set(gca,'Xlim',[xLB(1) xUB(1)],'Ylim',[xLB(2) xUB(2)]);
tightfigUA();
%%---END Depict neural implicit constitutives and aggregated constraint
%-END- Final Printing & Plotting


% % --- Compute perimeter & Area
[s] = plotcontour(C);
for k=1:size(struct2cell(s'),2)
[Perimeter_CA,Area_CA] = interpclosed(s(k).xdata,s(k).ydata,'pchip'); % only for contiguous contour
Perimeters(k)=Perimeter_CA; Areas(k)=Area_CA;
end
Perimeters
Areas
% %---END Compute perimeter & Area

% ------------------------------------------------------------------------

% ======
function [Obj NetOutX] = ObjMatNN(Pars)

global NetInput nNetOutput
global Rho iplot remplot NPts nPtsX nPtsY nMesh
global X Y X1 X2 C0

Pars=Pars(:);

%--- NNet
[NetOutX] = MatNNet(Pars,X);
[NetOutY] = MatNNet(Pars,Y);
%-END- NNet


%--- Constraint Aggregation
[CAX]=ConAgg(nNetOutput,NetOutX,Rho);
[CAY]=ConAgg(nNetOutput,NetOutY,Rho);
%-END- Constraint Aggregation

Obj = 1e+0*(sum(max(0,CAX)) + sum(max(0,-CAY))) + ...
1e+0*(sum(CAX > 0) + sum(CAY <= 0)) + ...
1e-5*norm(Pars,1);

%--- PLOT Intermediate
iplot=iplot+1;
if (mod(iplot,remplot)==0);

fig1=sfigure(1);
plot(X(1,:),X(2,:),'b.'); hold on; plot(Y(1,:),Y(2,:),'r.');
hold on
%--- Constraint Aggregation
```

```
% --- NNet
[NetOut] = MatNNet(Pars,NetInput);
% -END- NNet
[CA]=ConAgg(nNetOutput,NetOut,Rho);
Z=(reshape(CA,nMesh,nMesh));
%-END- Constraint Aggregation
plotcontour(C0);
[~,h] = contour(X1,X2,Z,[0 0],'k-','LineWidth',2); grid off;
set(h,'ShowText','off');,
title(['ObjCall# = ',num2str(iplot),'   ObjValue = ',num2str(Obj)])
xlabel('x_1','FontSize',12)
ylabel('x_2','FontSize',12)
movegui(fig1,'northeast');
drawnow
hold off
end
%-END- PLOT Intermediate

end


% ======
function [CA] = ConAgg(~,Cons,Rho)
maxtermCons = max(Cons,[],1);
CA = maxtermCons+log( sum( exp(Rho*(Cons-maxtermCons)), 1 ) )/Rho; % All must be <= 0 for X & >= 0 for Y
end



% ======
function [NetOut] = MatNNet(NetPars,NetInput)
global nNetOutput term1 term2 WI

% %--- NNet-Hidden Layer
WO=reshape(NetPars(1:term1),nNetOutput,term2);

term3=ones(1,length(NetInput));
ZI = [term3; NetInput];

ZIW = WI*ZI;
ZIWA = max(0,ZIW); % ReLU

% %---END-NNet-Hidden Layer

% %--- NNet-Output Layer
ZO = [term3; ZIWA]; % with Output Biasses
%ZO = [ZIWA]; % without Output Biasses
ZOW = WO*ZO;

NetOut = ZOW; % Linear
% %---END NNet-Output Layer
end
% ======
function [g] = constraints(x)

g(:,1) = (x(:,1)-7);
g(:,2) = (-x(:,1)+1);
g(:,3) = (x(:,2)-5);
g(:,4) = (-x(:,2));
g(:,5) = (x(:,1)+x(:,2)-10);
end
```

```
% ======
function [s] = plotcontour(c)
tol=1e-12;
k=1;    % contour line number
col=1;  % index of column containing contour level and number of points
while col<size(c,2); % while less than total columns in c
idx=col+1:col+c(2,col);
s(k).xdata = c(1,idx).';
s(k).ydata = c(2,idx).';
%   s(k).isopen = abs(diff(c(1,idx([1 end]))))>tol || ...
abs(diff(c(2,idx([1 end]))))>tol;
plot(s(k).xdata,s(k).ydata,'g-','LineWidth',2); hold on; % UA
k=k+1;
col=col+c(2,col)+1;
end
end
```