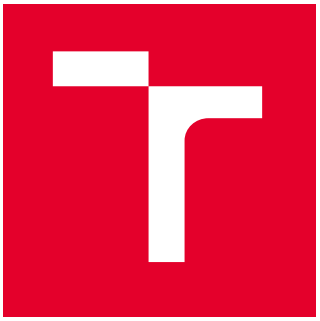

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV RADIOELEKTRONIKY

DEPARTMENT OF RADIO ELECTRONICS

OPTIMALIZACE ANTÉNY S KRUHOVOU POLARIZACÍ

CIRCULARLY POLARIZED ANTENNA OPTIMIZATION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Vojtěch Niederle**

AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Petr Kadlec, Ph.D.**

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Elektronika a komunikační technologie**

Ústav radioelektroniky

Student: Vojtěch Niederle ***ID:*** 221342

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Optimalizace antény s kruhovou polarizací

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s principy optimalizace pomocí tzv. Gaussovských procesů [1]. Implementujte tuto optimalizační techniku v prostředí MATLAB. Porovnejte konvergenci metody založené na Gaussovských procesech s tradičními stochastickými optimalizačními metodami (GA, PSO, DE aj.) na škálovatelných testovacích úlohách. Pomocí implementovaného optimalizačního algoritmu navrhněte anténu s kruhovou polarizací pro pásmo ISM 5.8 GHz. Návrh ověřte pomocí simulace v CST MWS či v HFSS. Anténu vyrobte a porovnejte měřené vlastnosti se simulovanými.

DOPORUČENÁ LITERATURA:

[1] HE, Zhenan a Gary G. YEN. LIU, Miao, Girish CHOWDHARY, Bruno CASTRA DA SILVA, Shih-Yuan LIU a Jonathan P. HOW. Gaussian Processes for Learning and Control: A Tutorial with Examples. IEEE Control Systems [online]. 2018, 38(5), 53-86 [cit. 2021-5-24]. ISSN 1066-033X. Dostupné z: doi:10.1109/MCS.2018.2851010.

[2] MILLIGAN, Thomas A. Modern antenna design. New York: McGraw-Hill, 1985.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 29.5.2023

Vedoucí práce: doc. Ing. Petr Kadlec, Ph.D.

doc. Ing. Lucie Hudcová, Ph.D.
předseda rady studijního programu

ABSTRACT

This thesis deals with optimization using Gaussian processes and differential evolution algorithm, and its implementation in MATLAB. The theoretical part discusses the topic of Gaussian processes in terms of mathematical representation, choice of kernel, parameters, mathematical procedures for implementation, differential evolution algorithm its mutation, crossover, and theory of circularly polarized antenna. The solution section describes the implementation in MATLAB and deals with the testing of this algorithm on benchmark functions. The results are then compared with other stochastic optimization algorithms and the effectiveness of the method is discussed for different types of problems. Based on the results, the method is extended by a Differential evolution algorithm, and circularly polarized antenna is optimized by this method. Designed antenna is fabricated and measured. The results are compared with simulations.

KEYWORDS

Gaussian process regression, squared exponential kernel, MATLAB, optimization, optimization testing functions, differential evolution algorithm, mutation, crossover, circularly polarized antenna, patch antenna

ABSTRAKT

Tato práce se zabývá optimalizací pomocí Gaussových procesů, algoritmu diferenciální evoluce a její implementací v prostředí MATLAB. Teoretická část se zabývá tématem Gaussových procesů z hlediska matematické reprezentace, volby jádra, parametrů, matematických postupů pro implementaci, algoritmem diferenciální evoluce, mutací, křížením a teorií kruhově polarizovaných antén. Praktická část popisuje implementaci v prostředí MATLAB a zabývá se testováním tohoto algoritmu na testovacích funkcích. Výsledky jsou pak porovnány s jinými stochastickými optimalizačními algoritmy a je diskutována efektivita metody pro různé typy problémů. Na základě výsledků je metoda rozšířena o algoritmus diferenciální evoluce a je provedena optimalizace kruhově polarizované antény. Navržená anténa je vyrobena a změřena, výsledky jsou porovnány se simulacemi.

KLÍČOVÁ SLOVA

Regresse Gausovských procesů,kvadratické exponenciální jádro, MATLAB, optimalizace, testovací funkce pro optimalizaci,algoritmus diferenciální evoluce, mutace, křížení, kruhově polarizovaná anténa, flíčková anténa

ROZŠÍŘENÝ ABSTRAKT

Tato práce se zabývá Gausovskými procesy $\mathcal{GP}$ a jejich využitím pro optimalizaci kruhově polarizované antény. $\mathcal{GP}$ je jedna z metod strojového učení, které je v dnešní době stále více rozšířné. $\mathcal{GP}$ mohou být použity nejen jako modely funkcí, kdy při zadání vstupních proměnných $\mathcal{GPR}$ vypočítá předpokládané výstupy, ale i jako optimalizační algoritmus. Optimalizační algoritmy obecně slouží k nalezení ideální kombinace proměnných pro požadovaný výsledek. Může se jednat o problém s velkým množstvím vstupních proměnných, jejichž kombinace není možná určit , nebo jejichž výpočet by byl složitý, nebo nepřesný. $\mathcal{GP}$ není přímo optimalizační metoda, ale je možné ji k optimalizaci využít. V tomto případě je výhodou, že při optimalizaci je zároveň trénován model problému. Tato práce se zabývá především Gausovskými procesy v regressi $\mathcal{GPR}$, která je použita pro trénování modelu.

Teoretická část práce popisuje princip $\mathcal{GPR}$. V první části teorie je popsána změna modelu $\mathcal{GP}$, v závislosti na počtu trénovacích dat. Následující část se zabývá kovarianční maticí $\mathbf{K}$, která udává závislosti jednotlivých proměnných vůči sobě. Důležitá je i volba správné kovarianční funkce, která ovlivňuje chování modelu. Na základě této matice je možné generovat náhodné funkce, které korespondují s trénovacími daty. Dále je popsán způsob výpočtu předpokládané hodnoty trénovací funkce μ. V této části práce je také popsána implementace $\mathcal{GP}$, pracujících s nepřesnostmi trénovacích dat.

V následující části teorie je popsán algoritmus diferencialní evoluce DEA. Tato část popisuje základní fungování algoritmu jako například mutace, křížení nebo selekce. Tento algoritmus je použit při optimalizaci pro dosažení lepších výsledků.

V poslední teoretické části je popsána problematika antén a to především kruhově polarizované flíčkové antény. Tato část se zaměřuje na základy návrhu flíčku, napájení a vysvětluje kruhovou polarizaci.

Optimalizační algoritmus je napsaný v programovacím jazyku MATLAB. Pro správné fungování optimalizátoru bylo třeba ošetřit zaokrouhlování. Při optimalizaci s velkým množstvím dat docházelo k drobným chybám ve výpočtu matic, což nevedlo k pozitivně definitní matici, která je vyžadována u Choleského dekompozice. Tato výjmka byla ošetřena přidáním kódu, který nalezne nejbližší pozitivně definitní matici. Tento algoritmus byl testován na čtyřech testovacích funkcích: Acklyho funkce, Dixon a Pricova funkce, Rastriginova funkce a sférická funkce. Testování bylo nastaveno na maximálně 1550 trénovacích dat. Trénovací data byla při optimalizaci získávana z testovacích funkcí. Tyto optimalizace byly provedeny pro dvou až pěti dimensionální problém. Pro objektivní porovnání byla každá optimalizace provedena stokrát a posuzován byl medián, minimum, maximum a směrodatná odchylka výsledků. Toto testování bylo provedeno i pro Algoritmus diferenciální evlouce DEA, Genetický algoritmus GA, Optimalizaci hejnem částic PSO a

Samosorganizující migrační algoritmus. V porovnání s těmito metodami jsou optmializace pomocí $\mathcal{GPR}$ horší , krom optimalizace sférické funkce. Další nevýhodou $\mathcal{GPR}$ jsou požadavky na výpočetní výkon a doba optimalizace. Tyto problémy jsou způsobeny náhodným výběrem testovacích dat, na základě kterých jsou volena data pro trénování. Tento přístup ke generaci nových trénovacích dat je především nevýhodný pro více dimenzionální problémy, kde není možné počítat s dostatečně velkým vzorkem testovacích dat.

Vzhledem k špatným výsledkům byl $\mathcal{GPR}$ algoritmus vylepšen pomocí DEA. Toto rozšíření spočívalo v hledání minima μ modelu $\mathcal{GP}$ pomocí DEA. Aby se zamezilo stagnování algoritmu v lokálním minimu, jsou počítány i hodnoty variance $\mathbb{V}$ pro náhodná testovací data. Touto úpravou bylo dosaženo výrazného snížení časů jednotlivých iterací. Tento rozšířený algoritmus byl testován na stejných testovacích funkcích jako algoritmus původní. V tomto případě byl maximální počet trénovacích dat snížen na 400. I přes toto snížení došlo k zelpšení výsledků Ackleyho funkce a sférické funkce. Výsledky testování na Rastriginově funkci byly obdobné jako u původního algoritmu. Vzhledem k těmto výsledkům je algortimus vhodný pro optimalizaci antény.

Pro optimalizaci antény je nejdříve nutné zvolit správný typ antény. Vzhledem k ceně výroby a vlastnostem je zvolena flíčková obdelníková anténa. Při volbě napájení je nutné zohlednit požadovanou šířku pásma a možnosti pro optimalizaci. Napájení je provedeno pomocí koaxiální sondy, a to vzhledem k dostatečným možnostem šířky pásma a k možnosti optimalizace pomocí polohy napaječe. Jako substrát je zvolen CuClad 217 s relativní permitivitou ε_{r}=2.2 o tloušťce 1.524mm. Při volbě tlustšího substrátu by se mohly projevit parazitní vlastnosti napaječe. Rozměry flíčku a poloha napaječe jsou zjištěny pomocí PSO v programu CST Studio Suite. Takto navržená anténa má činitel odrazu $S11$=-44,6 dB a osový poměr AR=0.314dB. Toto navržení mělo za cíl zjistit jaké parametry by měla mít optimalizovaná anténa.

Výsledky rozměrů antény, dosažené pomocí PSO, jsou použity pro stanovení oblasti, ve které bude anténa optimalizována. Limity testování optimalizace jsou stanoveny jako $\pm$10% rozměrů antény optimalizované pomocí PSO. Výsledky jednotlivých antén jsou zjednodušeny na kriteriální funkci vypočtenou z $S11$ a AR. Optimalizace pomocí $\mathcal{GPR}$ začíná vygenerováním 200 náhodných antén. Tyto antény jsou použity pro prvotní natrénování $\mathcal{GP}$. Poté je spuštěna optimalizace v průběhu které je simulováno 500 antén, na kterých je model průběžně testován. Strategie výběru trénovacích dat je stejná jako u testování na funkcích, 4 trénovací data jsou vybrány na zákaldě $\mathbb{V}$ a 1 data na základě minima μ. V tomto případě se ukázalo že algoritmus nedokáže jemně dolazovat ve velkých oblastech. Nejlepší výsledek byl dosažen již při 200 trénovacích funkcích. Toto může být způsebonu zaokrouhlováním při řešení lineárních rovnic, nebo velmi nízkou hodnotou $\mathbb{V}$ v oblasti řešení. Obdobná

optimalizace byla provedena ještě jednou a to pro limity ±2% nejlepšího řešení předchozí optimalizace. V tomto kroku již bylo dosaženo velmi nízké hodnoty kriteriálná funkce. Optimalizace byla spuštěna ještě jednou a to pro doladění řešení. Opět se jednalo zmenšení limit na ±2% dosavadního nejlepšího řešení. V tomto případě již nebyl model trénován na základě $\mathbb{V}$, ale pouze pro nejnižší hodnoty μ. Během této optimalizace bylo provedeno 101 simulací a bylo dosaženo relevantního výsledku. Celkový počet simulací pro optimalizaci je 1501. Optimalizovaná anténa byla téměř totožná s anténou optimalizovanou pomocí PSO. Anténa optimalizovaná pomocí $\mathcal{GPR}$ má činitel odrazu $S11$=-46,0 dB a osový poměr AR=0.160dB. Vyzařovací charakteristiky antény odpovídají RHCP. Pro objektivní zhodnocení výsledku by bylo lepší optimalizaci provést vícekrát, což nebylo možné kvůli velkému výpočetnímu výkonu.

Anténa je realizována vyleptáním flíčku do mědí pokrytém substrátu Arlon CuClad217 o výšce 1.524 mm. Napájení je provedeno SMA konektorem připájeným k flíčku a zemnící ploše. Měření charakteristik bylo provedeno v bezodrazové komoře a $S11$ parametru na vektorovém obvodovém analizátoru. Vyrobená anténa má posunutý rezonanční kmitočet f_0=5,93 GHz a AR=0,315dB. Změřené vyzařovací charakteristiky odpovídají simulacím. Vyrobená anténa je RHCP. Zisk antény je G= 7,5 dBi na f_0=5,93 GHz a G=6,85 dBi na f_0=5,8 GHz. Tento posun rezonanční frekvence může být způsoben menší reálnou hodnotou ε_r. Toto posunutí odpovídá simulaci s ε_r=2,1.

Optimalizace pomocí $\mathcal{GPR}$ je vhodná pro optamalizace antén. Jednou z výhod je možnost pozdějšího použití modelu natrénovaného při optimalizaci. $\mathcal{GPR}$ je výkonově náročnější oproti PSO, ale rychlost jednotlivých iterací lze ragulovat změnami parametru.

Bibliografická citace

NIEDERLE, Vojtěch. *Optimalizace antény s kruhovou polarizací* [online]. Brno, 2023 [cit. 2023-05-29]. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/151721. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky. Vedoucí práce Petr Kadlec.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Vojtěch Niederle*
VUT ID studenta:	*221342*
Typ práce:	*Bakalářská práce*
Akademický rok:	*2022/23*
Téma závěrečné práce:	*Optimalizace antény s kruhovou polarizací*

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 29. 5. 2023 ----------

podpis autora

ACKNOWLEDGEMENT

I would like to thank the advisor of my thesis, doc. Ing.Petr Kadlec, Ph.D. for his attitude, willingness and patience..

Contents

List of Figures

List of Tables

Listings

Introduction

As performance increases, price and the size of processors decreases, machine learning problem solving and optimization are becoming more common and as a result, many more resources are being invested in this sector. Today, machine learning and optimization are used not only in robotics but also in translators, search engines, statistics, and many others [1].

This thesis will mainly discuss optimization by Gaussian process regression to design an algorithm that will be able to find the optimum based on known samples. Based on the measured samples this method predicts the function and confidence interval at any point to help choose the next sample point. This optimization method can be particularly advantageous if the sampling data is obtained by a power-intensive simulator or if it is obtained from physically realized samples. Especially for many input variables, where more problem dimensions require many more samples, it pays off to use the algorithm to optimally select new sampling data.

One of the problems for which can be advisable to use an optimizer may be a circularly polarized antenna. The characteristics of a circularly polarized antenna include the axial ratio and the $S11$ parameter, which can be achieved by proper antenna design [2].

Aim of the Thesis

This thesis aims to theory and implementation of optimization using Gaussian process regression to design the circularly polarized antenna optimized for operating frequency of 5.8GHz.

As for the Gaussian process optimization, the goal is to write a code in MATLAB using Gaussian process regression to calculate the value and confidence interval at each point from sampling data, and automate this method so that the program will automatically select the next sampling data based on the Gaussian process results.

The other part of this thesis is to let other optimization methods find optimums of testing functions and compare results with the Gaussian process optimization algorithm written in the previous assignment. Only stochastic algorithms will be used here such as Genetic algorithm, Particle swarm optimization, Self organizing migrating algorithm and Differential evolution algorithm.

In the last part of the thesis, the algorithm is used to optimize a circularly polarized antenna for a frequency of 5.8GHz. After that, the thesis is focused on the antenna realization, its measurement, and the comparison of the measured values with the simulation.

1 Theory

1.1 Gaussian Process Regression

1.1.1 Theoretical Background

The first section will discuss inputs and outputs of Gaussian process regression $\mathcal{GPR}$ and what their practical use is.

Gaussian process calculates with two main input dataset testing data x_* and training data $f_t(x_t)$ and returns mean μ and variance $\mathbb{V}$ that will be used for the optimization algorithm. For better understanding $f_t(x_t)$ are measured data, μ are predictions, and $\mathbb{V}$ are predicted variances. Mean μ and variance $\mathbb{V}$ are based on $f_t(x_t)$ and can be calculated at any position x_*. For example, let´s say that there are some measured points $f_t(x_t)$ from unknown problem. Let's consider the problem: $(6x-2)^2 sin(12x-4)$ [3]. Testing data x_* are large finite dataset of points in interval $x_* \in [0,1]$ [3]. By applying $\mathcal{GPR}$ values μ and $\mathbb{V}$ are calculated for every value x_* as shown in Fig. 1.1.

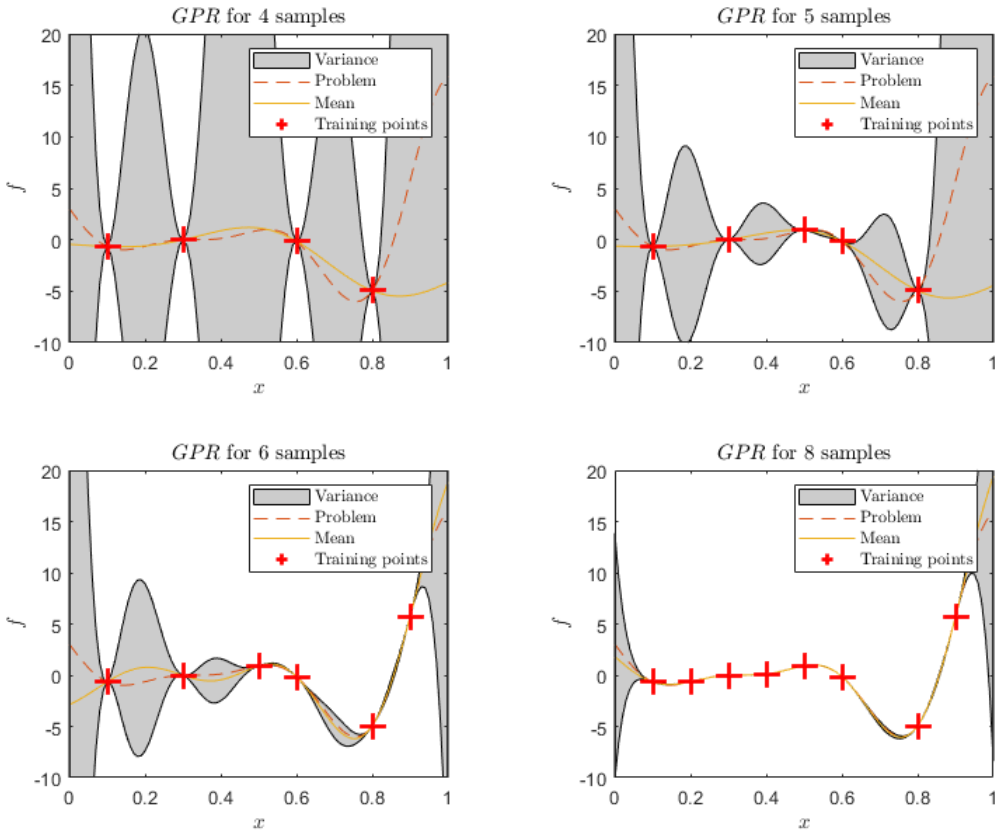

Fig. 1.1: $\mathcal{GPR}$ example for 4, 5, 6 and 8 samples

As shown in Fig. 1.1 predicted function μ and variance $\mathbb{V}$ can predict next x_t. The biggest priority is to cover points with high $\mathbb{V}$ to have sufficiently precise μ. The required precision depends on the character of the unknown function. Functions with a lot of local extremes require lower $\mathbb{V}$. Another approach for choosing the next x_t can be to choose points near to minimum of μ. Choosing the next sample based on the position of the smallest μ can refine the local minimum position, but can't find any new minimum located far from this point.

Another important variable is noise. Fig. 1.1 shows noiseless data $f_\mathrm{t}(x_\mathrm{t})$ and the noiseless $\mathcal{GPR}$ method. To work with data containing noise, the method has to be modified and an input variable indicating expected noise must be added. This thesis deals primarily with noise-less data. The goal of the thesis is to test $\mathcal{GPR}$ method on noiseless training functions and work with noiseless samples obtained from simulator.

1.1.2 Kernel Matrix

One of the essential parts of the $\mathcal{GP}$ is a kernel matrix. In some literature, it is referred to as covariance matrix [4]. Kernel matrix $\mathbf{K}(x_1, x_2)$ declares the similarity of each point from dataset x_1 to each point in dataset x_2. For most kernels, the similarity is usually described in the range zero to one where one is the highest similarity and zero is the lowest. To be more specific, the structure of kernel matrix $\mathbf{K}$ is described by the equation [1]:

$$\mathbf{K}(x_1, x_2) = \begin{bmatrix} \mathbf{k}(x_{11}, x_{21}) & \mathbf{k}(x_{11}, x_{22}) & \dots & \mathbf{k}(x_{11}, x_{2j}) \\ \mathbf{k}(x_{12}, x_{21}) & \mathbf{k}(x_{12}, x_{22}) & \dots & \mathbf{k}(x_{12}, x_{2j}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{k}(x_{1i}, x_{21}) & \mathbf{k}(x_{1i}, x_{22}) & \dots & \mathbf{k}(x_{1i}, x_{2j}) \end{bmatrix} \tag{1.1}$$

where $\mathbf{k}(x_{1i}, x_{2j})$ is the covariance function. There are different covariance functions but most of them are based on the distance between two points $\boldsymbol{d}$. The i is index of a point in dataset x_1 and j is index of the point in dataset x_2. This can be also applied to multivariate x but then $\boldsymbol{d}$ must be calculated using the Euclidean distance between the points:

$$\boldsymbol{d}(x, y) = \sqrt{\sum_{n=1}^{D} (y_D - x_D)^2} \tag{1.2}$$

As already mentioned, every value in the kernel matrix is the Euclidean distance recalculated using covariance function. There are a few covariance functions, the differences between them are mainly in the resulting μ and $\mathbb{V}$. Square exponential [4]:

$$\mathbf{k}(x_1, x_2) = exp\left(-\frac{\boldsymbol{d}(x_1, x_2)^2}{2l^2}\right) \tag{1.3}$$

γ-exponential [4]:

$$\mathbf{k}(x_1, x_2) = exp\left(-\left(\frac{\boldsymbol{d}(x_1, x_2)}{l}\right)^{\gamma}\right) \tag{1.4}$$

Rational quadratic [4]:

$$\mathbf{k}(x_1, x_2) = \left(1 + \frac{\boldsymbol{d}(x_1, x_2)^2}{2\alpha^2}\right)^{-\alpha} \tag{1.5}$$

where l, γ, and α are parameters that change the character of the kernel. Often the covariance function is multiplied by the constant [4]. This leads to an increase in the amplitude of variance $\mathbb{V}$. Fig. 1.2 shows the kernel matrices with different covariance functions. The i and j axis describe the position of points in datasets x_1 and x_2, respectively x_1 and x_2 are linear distributions from -1 to 1 containing 100 values. This means that at the position 1,1 Fig. 1.2 the value is 1 because the similarity of the first point from dataset x_1 and the first point from dataset x_2 is absolute since datasets x_1 and x_2 are equal. As illustrated Fig. 1.2 the covariance function indicates how the method will behave with regard to the distance between the points. Kernel matrices are always symmetric positive definite which is also a condition for the Cholesky decomposition which will be discussed later in the thesis [1].

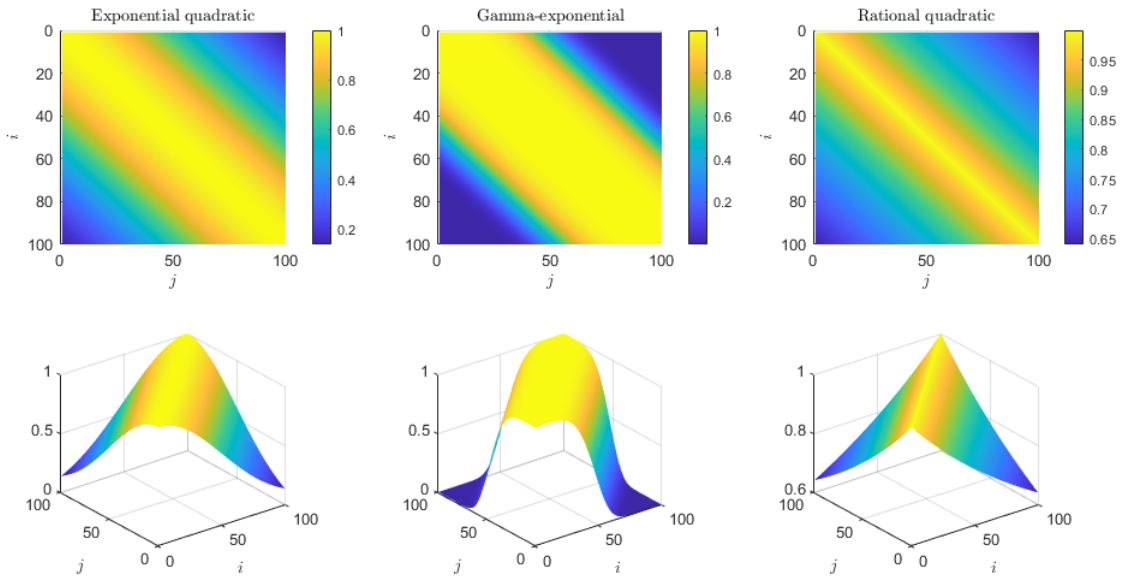

Fig. 1.2: Kernel matrices with diferent covariance functions

The square exponential covariance function is referred to as one of the most commonly used in open literature [4]. The thesis mainly deals with this covariance function. Covariance functions above have a possible limitation and that is the value of $\mathbb{V}$. As the parameter l increases, the value of $\mathbb{V}$ decreases significantly and vice-versa. This can lead to calculation problems caused by rounding. The most common method to avoid this is to multiply the function by a parameter σ as follows [5]:

$$\mathbf{k}(x_1, x_2) = \sigma^2 exp\left(-\frac{\boldsymbol{d}(x_1, x_2)^2}{2l^2}\right) \tag{1.6}$$

As shown in Fig. 1.3 and Fig. 1.4, the parameter l determines the horizontal scale and σ the vertical scale. When l is low, μ drops to zero, and in this region $\mathbb{V}$ is constant. But if l is high, $\mathbb{V}$ is very low so μ is directed according to the directive given by the training points. As for the shape of $\mathbb{V}$, a higher value of l makes $\mathbb{V}$ sharper.

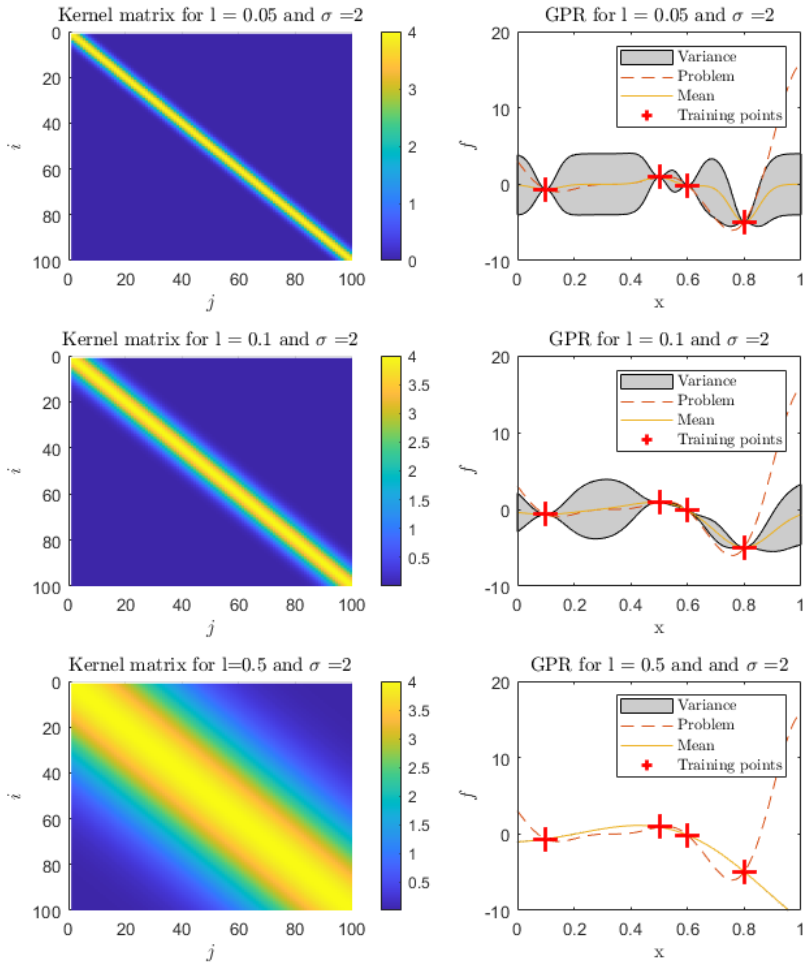

Fig. 1.3: Effect of variables l and σ on $\mathcal{GPR}$

Fig. 1.4 shows the shape of $\mathbb{V}$ at higher l by increasing σ. As shown in Fig. 1.3 small value of parameter l=0.05 is causing faster decreasing of values further from the diagonal. This means that points further apart from each other have much higher variance for smaller l. If l is high, the values decrease more slowly as they move away from the diagonal. This means that points $f_{\mathrm{t}}(x_{\mathrm{t}})$ need to be further apart to increase the $\mathbb{V}$. The σ parameter increases the scale of kernel matrix which causes the amplitude of the variance to increase while keeping its shape and no change of μ.

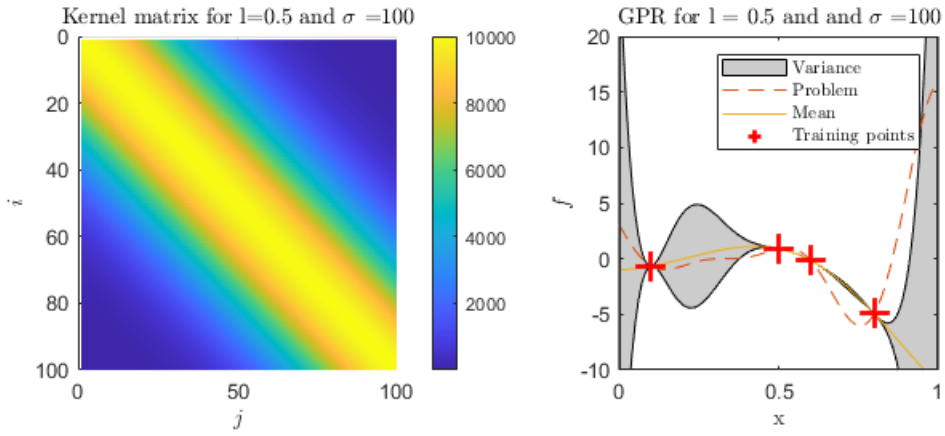

Fig. 1.4: Effect of l on $\mathcal{GPR}$

1.1.3 Getting Random Functions from $\mathcal{GPR}$

With knowledge of the kernel matrices, it is possible to make equations for mean and variance. In the literature, the Gaussian process is described as the distribution over function [1]. This means that it is possible to create infinitely many functions corresponding to sampled points. The left side of Fig. 1.5 shows five random functions generated with mean 0. The right side of Fig. 1.5 shows the random functions generated with μ and $\mathbb{V}$ function obtained from the Gaussian process with four training points. Random function with acording to covariance matrix [4]:

$$f \sim \mathcal{N}(\mu, \mathbf{K}) \begin{cases} \mu = [0, 0, 0, ..., 0] & \text{length of } \mu \text{ is equal to length of x} \\ \mathbf{K} = \mathbf{K}(x, x) & \text{covariance matrix of x points} \end{cases} \tag{1.7}$$

As shown on the right side of Fig. 1.5 all random functions cross the sampled points. That is due to the covariance matrix and mean. The covariance matrix declares low variance at samples position, which affects these random functions to approach the mean at the position. Mean function is created by $\mathcal{GPR}$ and it is

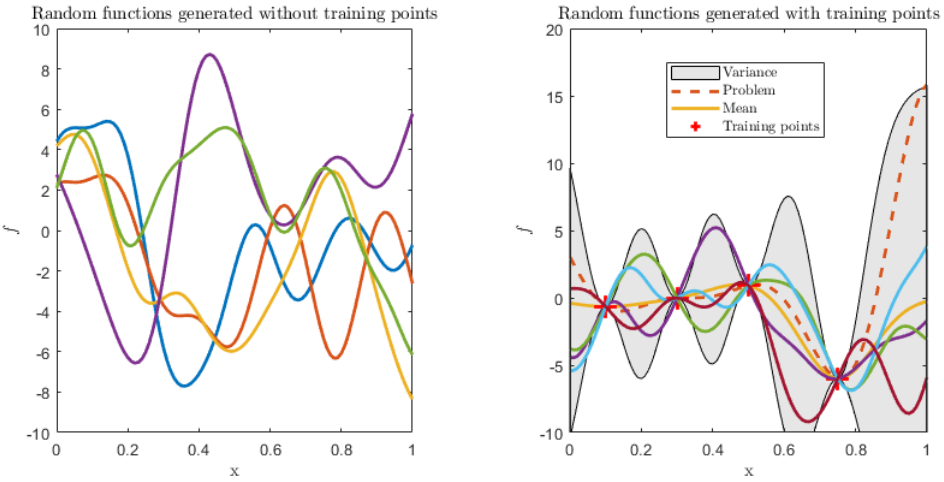

Fig. 1.5: Random functions

crossing the samples. Therefore, if there is low variance in the sample region and at the same time μ intersects the samples, the random functions will also cross the samples. The opposite situation occurs between the two samples. Here the variance is high so random functions can get further from the mean. In general, the larger the variance, the more random the function is at the area.

It can be noted that the term covariance matrix is used here. This different designation is because the covariance matrix does not directly declare the similarity between points as kernel matrix. The covariance matrix is the difference between the kernel matrix of x_* and kernel describing sampled points. Therefore the covariance matrix does not describe similarity but confidence in each point as shown in Fig. 1.6. This will be better discussed in the next section.

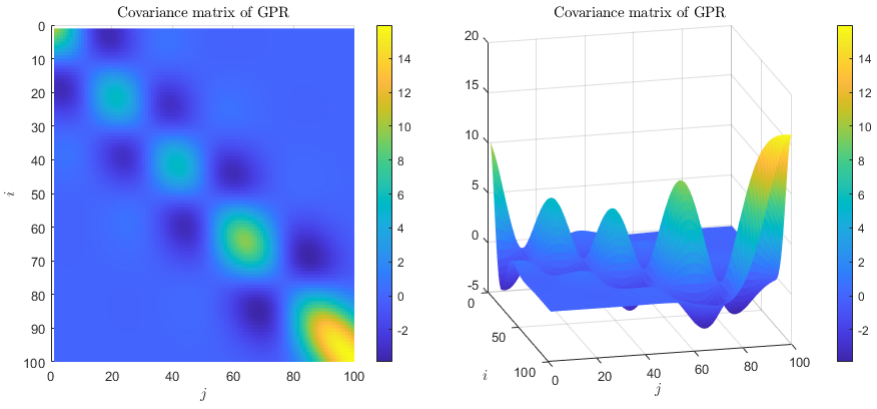

Fig. 1.6: Covariance matrix

1.1.4 Noiseless Gaussian Process Regression

As previously mentioned, the Gaussian process is declared as f from previous subsection. The other requirement to get random functions is to know the mean and covariance matrix. The subsection will deal with getting mean and covariance matrix for noiseless multivariate data. To express the formula of μ and $\mathbb{V}$ from the equation [7]:

$$f' \sim \mathcal{N}(\mu', \mathbf{K}') \tag{1.8}$$

The equation needs to be more specified. The f' represents the points, so the f' will be declared for testing inputs and training points as [6],[7]:

$$f' = \begin{bmatrix} f_{\mathrm{t}}(x_{\mathrm{t}}) \\ f(x_*) \end{bmatrix} \tag{1.9}$$

The mean μ' is equal to:

$$\mu' = \begin{bmatrix} \mu_* \\ \mu \end{bmatrix} \tag{1.10}$$

where μ_* is supposed mean and μ is mean function [6],[7]. The μ_* is the expected center of $f_{\mathrm{t}}(x_{\mathrm{t}})$ but it is usually considered as $\mu_* = 0$ [7]. The $\mu_* = 0$ is not recommanded for functions that have $f_{\mathrm{t}}(x_{\mathrm{t}})$ far from the zero because the μ tend to approach the μ_*. The covariance matrix contains kernel matrices with all combinations of input data [6],[7]:

$$\mathbf{K}' = \begin{bmatrix} \mathbf{K}(x_{\mathrm{t}}, x_{\mathrm{t}}) & \mathbf{K}(x_{\mathrm{t}}, x_*) \\ \mathbf{K}(x_*, x_{\mathrm{t}}) & \mathbf{K}(x_*, x_*) \end{bmatrix} \tag{1.11}$$

The $\mathbf{K}(x_*, x_*)$ is kernel matrix describing the similarity between x_* and x_* datasets, the $\mathbf{K}(x_*, x_{\mathrm{t}})$ between x_{t} and x_* etc. The $\mathbf{K}(x_{\mathrm{t}}, x_*)$ is equal to $\mathbf{K}(x_*, x_{\mathrm{t}})^T$ and for simplicity will be used in the further equations. The final equation may look like this [6],[7]:

$$\begin{bmatrix} f_{\mathrm{t}}(x_{\mathrm{t}}) \\ f(x_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_* \\ \mu \end{bmatrix}, \begin{bmatrix} \mathbf{K}(x_{\mathrm{t}}, x_{\mathrm{t}}) & \mathbf{K}(x_*, x_{\mathrm{t}})^T \\ \mathbf{K}(x_*, x_{\mathrm{t}}) & \mathbf{K}(x_*, x_*) \end{bmatrix} \right) \tag{1.12}$$

From that equation, it is possible to make conditions for distribution [6],[7]:

$$\mu = \mathbf{K}(x_*, x_{\mathrm{t}})^T \mathbf{K}(x_{\mathrm{t}}, x_{\mathrm{t}})^{-1} f_{\mathrm{t}}(x_{\mathrm{t}}) \tag{1.13}$$

$$\mathbf{K}' = \mathbf{K}(x_*, x_*) - \mathbf{K}(x_*, x_{\mathrm{t}})^T \mathbf{K}(x_{\mathrm{t}}, x_{\mathrm{t}})^{-1} \mathbf{K}(x_*, x_{\mathrm{t}}) \tag{1.14}$$

With μ and $\mathbf{K}$', the random functions from $\mathcal{GPR}$ can be obtained. The mean μ function Fig. 1.4 is a function given by μ and x_* , where μ are values of the y-axis and x_* are values of the x-axis. The variance in Fig. 1.4 is μ - $diagonal(\mathbf{K}')$ and μ + $diagonal(\mathbf{K}')$ for the y-axis and x_* for the x-axis.

The prove that the equations work is when x_*=x_t then the first part of the $\mathbf{K}$' would be shorted, leaving:

$$\mathbf{K}' = \mathbf{K}(x_*, x_*) - \mathbf{K}(x_*, x_\mathrm{t}) \tag{1.15}$$

and the variance function will equal zero. In other words, if the positions of the samples are the same as the positions at which the mean is calculated, the mean will be the same as the function given by the samples and variance will be sorted to zero.

1.1.5 $\mathcal{GPR}$ with Samples Containing Noise

The previous method dealt with noise-free samples and the predicted mean function intersected all samples. However, this can be a problem if the obtained samples are not completely accurate as shown in Fig. 1.7 on the left. Therefore, it is necessary to ensure that the mean function does not have to intersect the samples but will still consider the position of the points as shown in Fig. 1.7 on the right. The noise

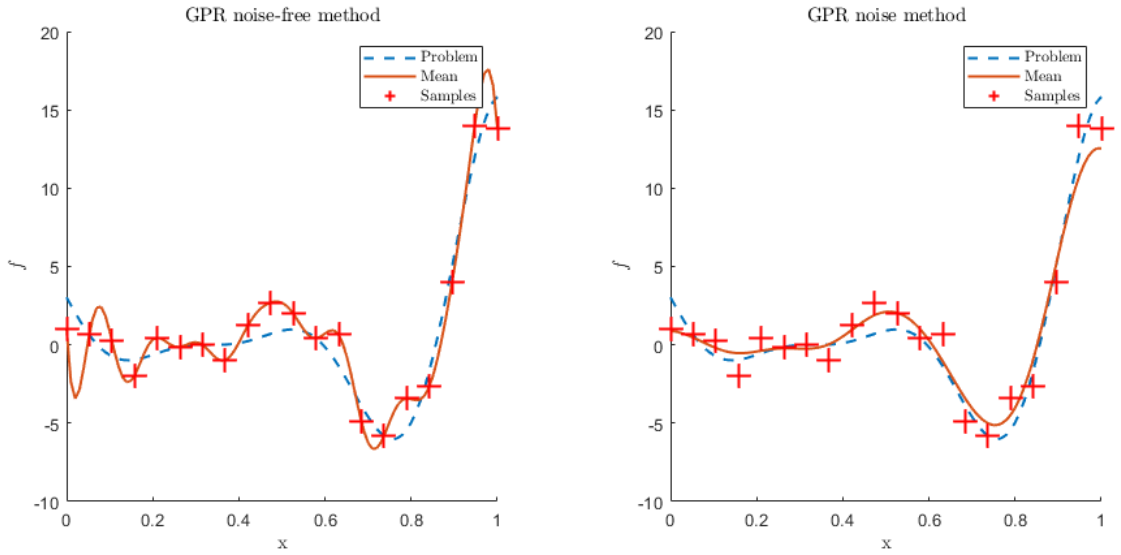

Fig. 1.7: $\mathcal{GPR}$ from samples containing noise

ϵ is equal to $\sigma^2 I$ where σ^2 is a matrix with ones on diagonal and I is amplitude parameter for noise. [7]:

$$\epsilon \sim \mathcal{N}(0, \sigma^2 I) \tag{1.16}$$

Then the $\mathcal{GPR}$ with noise is equal to [6],[7]:

$$f' \sim \mathcal{N}(\mu', \mathbf{K}' + \sigma^2 I) \tag{1.17}$$

$$\begin{bmatrix} f_{\mathrm{t}}(x_{\mathrm{t}}) \\ f(x_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_* \\ \mu \end{bmatrix}, \begin{bmatrix} \mathbf{K}(x_{\mathrm{t}}, x_{\mathrm{t}}) + \sigma^2 I & \mathbf{K}(x_*, x_{\mathrm{t}})^T \\ \mathbf{K}(x_*, x_{\mathrm{t}}) & \mathbf{K}(x_*, x_*) \end{bmatrix} \right) \tag{1.18}$$

Then the final conditions are [6],[7]:

$$\mu = \mathbf{K}(x_*, x_{\mathrm{t}})^T (\mathbf{K}(x_{\mathrm{t}}, x_{\mathrm{t}}) + \sigma^2 I)^{-1} f_{\mathrm{t}}(x_{\mathrm{t}}) \tag{1.19}$$

$$\mathbf{K}' = \mathbf{K}(x_*, x_*) - \mathbf{K}(x_*, x_{\mathrm{t}})^T (\mathbf{K}(x_{\mathrm{t}}, x_{\mathrm{t}}) + \sigma^2 I)^{-1} \mathbf{K}(x_*, x_{\mathrm{t}}) \tag{1.20}$$

Fig. 1.9 demonstrates the influence of parameter I to mean function. With lower I the mean function is getting closer to each sample. The higher value of I is making mean function to be less responsive to the samples.

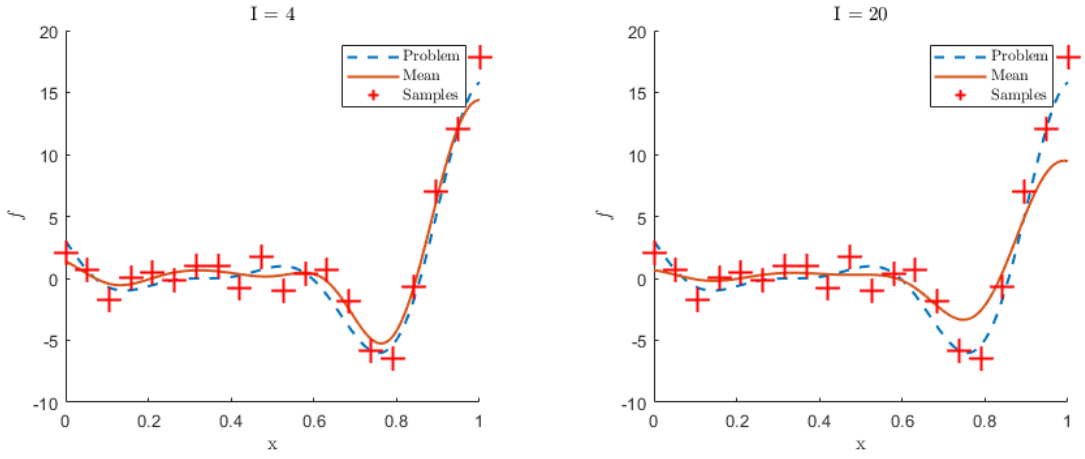

Fig. 1.8: Effect of noise variable I

1.2 Differential Evolution Algorithm

The Differential Evolution DE is a commonly used and versatile optimization method [9]. The DE is population base algorithm, which means that the algorithm is working with agents, that are stochastically renewed depending on their fitness error. The algorithm can be multi-objective or single-objective [11]. Only single-objective DE is discussed in the thesis, as all outputs of a problem will be recalculated into a single value. The DE consists of four main parts: initialization, mutation, crossover, and selection [8]. The initialization randomly creates a defined amount of parent vectors in a defined interval and their fitness function is calculated. The next part is the mutation, where trial vectors are created by stochastic recalculating from parent vectors. In the crossover part, some of the trial vector values are swapped between agents. The last part is selection. In this part of method, fintess errors of trial vectors are calculated. Trial vectors are compared with parent vectors and vectors with better fitness values will be used as a parent vectors. Entire process is repeated until the desired result is achieved.

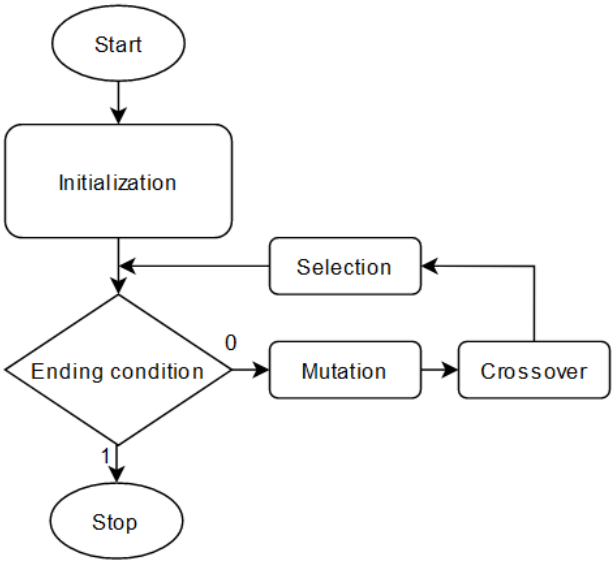

Fig. 1.9: Flowchart of DE

1.2.1 Initialization

As mentioned in the previous subsection, the initialization part is used for generation parent vectors. The parent vectors are generated by the following equation [8]:

$$x_{j,i} = rand_j(0,1)(ub_j - lb_j) + lb_j \tag{1.21}$$

where $x_{j,i}$ is a j-th dimension of an i-th agent, $rand_j(0,1)$ is a vector of random numbers from zero to one, ub_j and lb_j are lower and upper bounds of $x_{j,i}$ limits. Then, a fitness error is computed for each agent. The choice of the number of agents crossover ratio CR and scaling factor F can also be considered as part of the initialization. In the thesis these parameters are user-defined, self-adaptive DE can choose these parameters themselves during optimization [10].

1.2.2 Mutation

Once the agents are generated and the fitness error is calculated for them, the next step is to acquire new positions. For each agent, a new mutated vector $v_{j,i}$ is calculated based on the equation [8]:

$$v_{j,i} = x_{r0,i} + F[x_{r1,i} - x_{r2,i}] \tag{1.22}$$

where j is an index of new vector, i is an index of dimension, and $r0, r1$ and $r2$ are indexes of random parent vecor. In this case the $r0 \neq r1 \neq r2 \neq j$ but $r0$ can also be chosen based on another strategy [8]. The scaling vector is usually considered as $F = 1.5$ [11].

1.2.3 Crossover

The next part of the differential evolution algorithm is the crossover. Crossover ensures the creation of a new vectoru $u_{j,i}$ based on the combination of vectors $v_{j,i}$ and $x_{j,i}$. The following equation describes the choice of value of $u_{j,i}$ [8]:

$$u_{j,i} = \begin{cases} v_{j,i} & if(rand_j(0,1) \leq CR \text{ or } j = j_{rand}) \\ x_{j,i} & \text{else} \end{cases} \tag{1.23}$$

For every dimension j, the random number $rand_j(0,1)$ in range 0 to 1 is generated and for every agent the j_{rand} is generated in range of dimensions quantity. With the conditions above, the every new trial vector $u_{j,i}$ have at least one dimension paramterer from new vector $v_{j,i}$ thanks to j_{rand}. The other crossings are determined by CR.

1.2.4 Selection

The last part is selection. After calculating fitness error of trial vectors $u_{j,i}$ the population $x_{j,i}$ is updated. If the fitness error value of the vector $u_{j,i}$ is smaller than of vector $x_{j,i}$, the original vector $x_{j,i}$ is replaced by the vector $u_{j,i}$. Than the process is repeated from the mutation section until the optimization is complete.

1.3 Circularly Polarized Antenna

Circularly polarized antennas are used in applications where it is not possible to provide identical rotation of receiver and transmitter, often in applications where the receiver or transmitter is moving. These applications include, for example, aerospace, satellite, communications, navigation, or wireless sensors. One of the advantages is that they reduce the Faraday rotation effect, which would cause a signal loss in linearly polarized antennas [14]. As this effect is also caused by the ionosphere, circularly polarized antennas are commonly used in satellite systems. There are many designs of these antennas such as microstrip dipoles loops horns and many others [2].

1.3.1 Microstrip Antenna

Microstrip antenna is one of many designs to radiate or recieve an electromagnetic waves used in wireless communication. This type of antenna is widely used for its low cost combined with small dimensions (especially the profile) and low weight [12]. One of the many advantages of this antenna type is the ability to be easily optimized due to its simple design.

The design is based on a conductive patch placed on a dielectric substrate and a ground plane on the other side also from a conductive material. The patch can have different shapes such as rectangular, elliptical, triangular, or more complex shape. These shapes can be made by applying a thin layer of copper on a substrate, or by placing a thin copper plate on a substrate. This type of antenna may also be made of several layers of substrate stacked on each other. [2]. Dimensions of the patch are one of the main properties determining the characteristics of the antenna. Other things that affect the characteristics are relative permitivitty ε_{r} and thickness of a substrate. It is common to choose a substrate between ε_{r} values of 2.2 and 12, but it should be taken into consideration that with increasing permittivity the gain will decrease and the patch will need to be larger [13].

When choosing an antenna, it is necessary to target the attributes required by the application. These properties include dimensions, resonant frequency, radiation patterns, polarization, reflection coefficient, and bandwidth. Generally due to the design these antennas can be considered as low profile. Another essential parameter is the reflection coefficient Γ, which is related to operating frequency and bandwidth of the antenna. A patch antenna can have several operating frequencies on which it is able to efficiently transmit or receive a signal. One of the other criteria is the direction from which the antenna can receive the signal. These attributes are described by radiation patterns and polarization type.

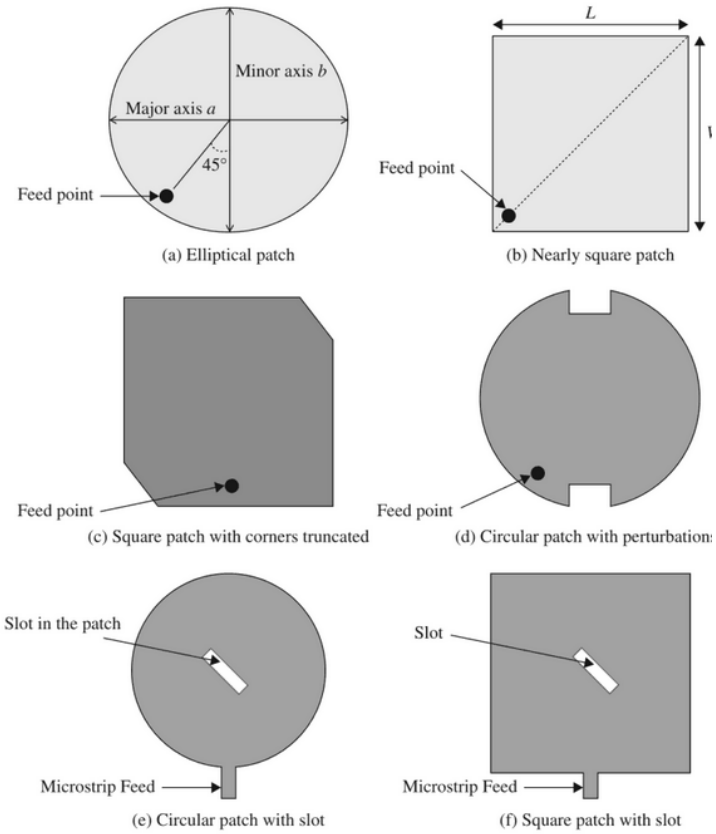

Fig. 1.10: Different patch shapes [14].

1.4 Circular Polarization

The basic types of antenna polarization are linear and circular. Unlike linear antennas, the plane of waves radiation rotate Fig. 1.11. Due to this, it is possible to transmit the signal without a significant gain drop when the rotation of a transmitter and a receiver are different. Circular polarization is further classified as right hand circular polarization $RHCP$ and left hand circular polarization $LHCP$ depending on the direction of rotation of the plane of wave propagation. It should be taken into consideration that in the case of $LHCP$, the antenna does not receive well the $RHCP$ signal and vice versa.

The level of circular polarization is given as axial ratio AR. This parameter indicates the difference between the largest and smallest value of the antenna radiation depending on the rotation of the measured and measuring antennas against each other. Since we aim for the gain to be independent of the rotation of the reciever and transmitter, the ideal value of AR is 0dB. Polarization can also be represented by a polarization ellipse which directly gives the rotation and radiation dependence Fig. 1.12.

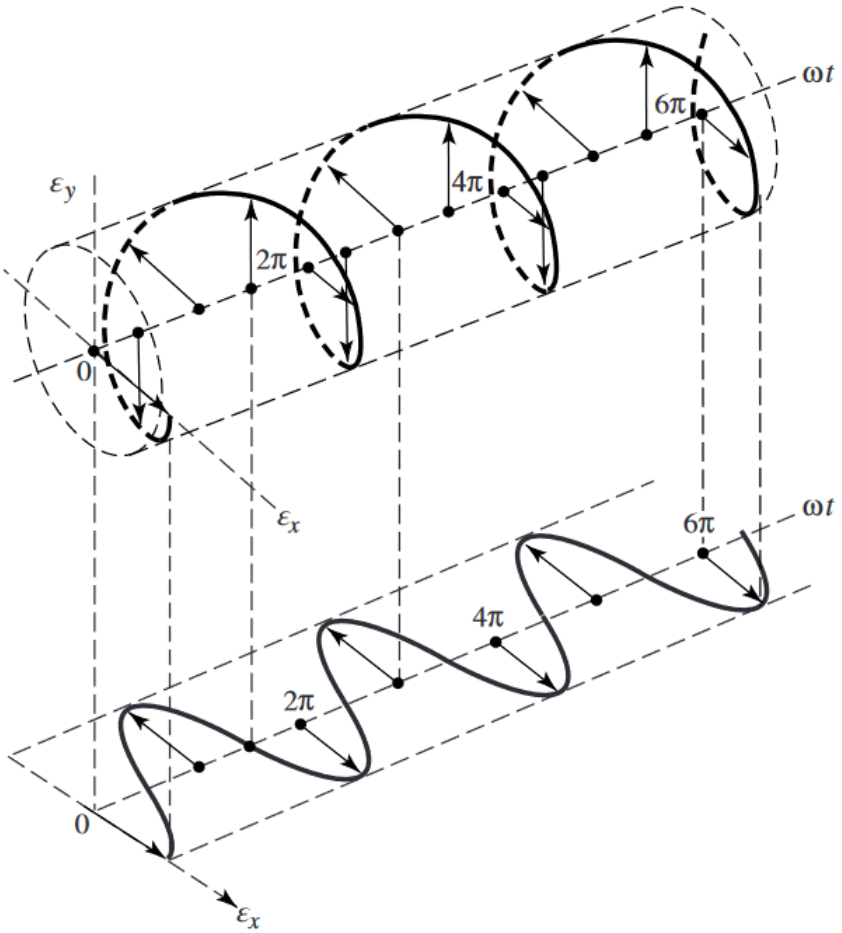

Fig. 1.11: Rotation of circularly polarized wave [13].

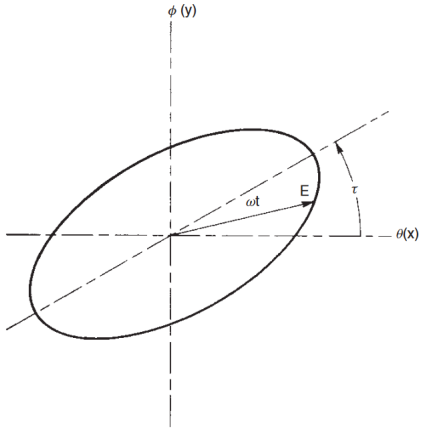

Fig. 1.12: Polarization ellipse [2].

Circular polarization can be achieved in several ways. One way is to have an asymmetrical position of the feed, or an asymmetrical antenna shape. An asymmetrical shape can be achieved by a cutout of patches corners Fig. 1.10 or by a completely asymmetrical design. Another way is by feeding the patch with multiple feeds with a phase shift. Another way is by using a cutoff in the middle of the antenna. Some antennas can be used as both $RHCP$ and $LHCP$ due to hybrid feeding Fig. 1.13.

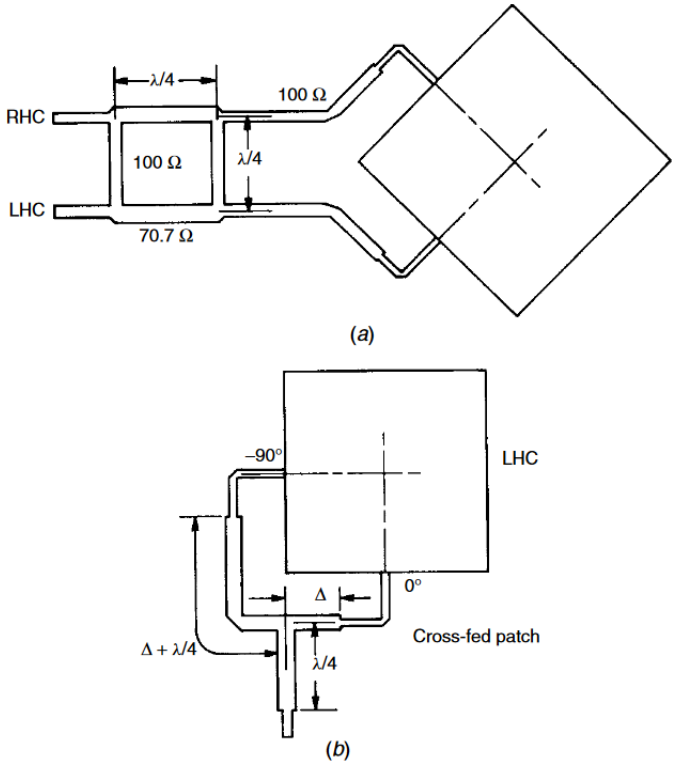

Fig. 1.13: Hybrid feed and dual feed antenna [2].

1.5 Antenna Feeding Techniques

When designing an antenna, it is important to correctly select the type of feed that is suitable for the application. The selected feed must be properly impedance matched to minimize wave reflections inside antenna. Feeders are normally designed with 50ohm impedance. The basic feeding techniques include microstrip line feed, coaxial promp feed, aperture-coupled feed and proximity-coupled Microstrip Feed [15]. Another thing that the feeding method affects is the polarization.

1.5.1 Microstrip Line Feed

This feed method consists of feeding electromagnetic waves into a patch using a microstrip line that is on a substrate connected to a patch. This type of feed have the advantage in simplicity of manufacture. In this type of feed it is nessesary to match the impednaces. The impedance matching of this type of feed is achieved by increasing the width of the feed or by inserting the feed into the patch. In this case, circular polarization can be achieved by creating asymmetries in the patch, or by dual supply with phase shift of 90 degrees Fig. 1.13.

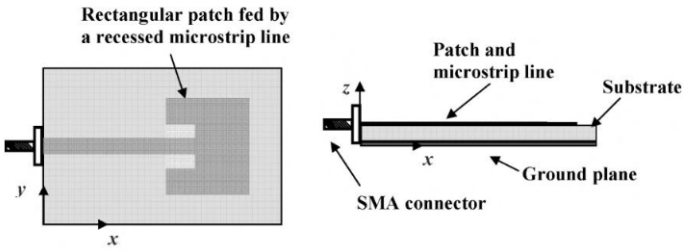

Fig. 1.14: Microstrip line feed [15].

1.5.2 Coaxial Probe Feed

Antenna feed by coaxial probe is realized from the bottom side of an antenna where the power supply is led through the substrate to a patch. This method of power supply is the most commonly used [15]. Advantages of this power supply include low radiation loss and the ability to feed the patch at any location [16]. Because of these advantages, this feeder is suitable for circularly polarized antennas, since by eccentrically positioning the feeder, circular polarization can be achieved. Disadvantages of this feeder include less bandwidth or more inductive character with a thicker substrate [16].

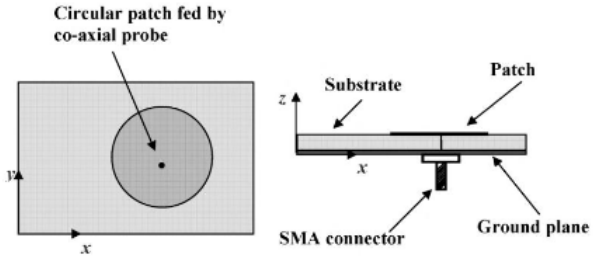

Fig. 1.15: Coaxial probe feed [15].

1.5.3 Aperture-Coupled Microstrip Feed

This method of indirect feeding consists of two layers of substrate, with a microstrip line on the bottom of the substrate and a ground plane between the substrates with an aperture in the middle. Advantages of this method of feeding include greater bandwidth and, compared to the microstrip line feed, limiting the radiation directly from the feeder [15]. Other advantages include no soldering. One of the disadvantages is the radiation from the back side of the antenna [15].

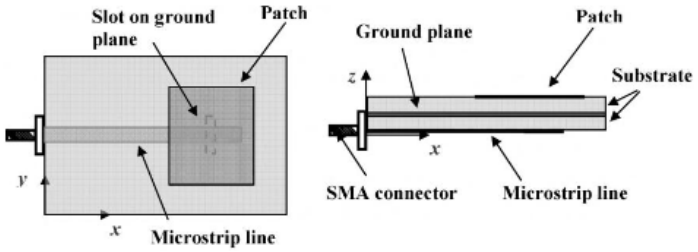

Fig. 1.16: Aperture-Coupled Microstrip Feed [15].

1.5.4 Proximity-Coupled Microstrip Feed

The design of this indirect feed method consists of two layers of a substrate on top of each other with a microstrip feed line located between the substrates. Compared to aperture-coupled microstrip feed, there is less back radiation of feed and wider bandwidth compared to other mentioned feeding techniques [15].

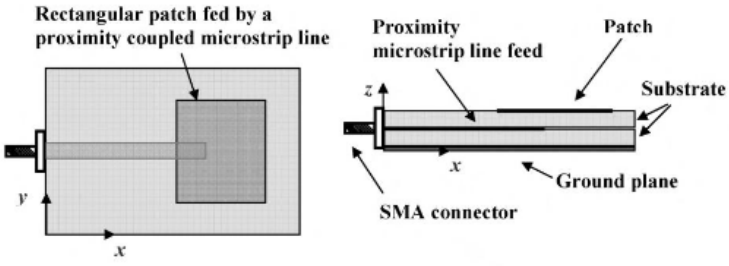

Fig. 1.17: Proximity-Coupled Microstrip Feed [15].

2 Implementation

2.1 Functions Requred for $\mathcal{GPR}$

The program is completely written in MATLAB 2021b. It contains five main files and a set of test functions.

One of the main files is the *expquadkernel.m* which is the function containing the square exponential covariance function. Defining the covariance function is also possible by using the function handle directly in the code.

Another important part of the code is the $euclideanDistanceBetweenTwoSets.m$ [18]. This function has two input datasets whose size is equalt to nomber of points. The output is the Euclidean distance matrix.

For proper functioning of the code, it is also necessary to take into consideration that the matrix is inverted using Cholesky decomposition which requires the matri to be positive definite [19]. The GPR method itself always generates a positive definite matrix but the rounding errors in MATLAB may occure. Therefore, it is necessary to use the $nearestSPD$ function that ensures the matrix is positive definite [20]. This function finds the nearest positive definite matrix wich can lead to a small error when evaluating new points, but in the vast majority of applications, it does not cause any noticeable decrease of the method efficiency.

2.2 The $\mathcal{GPR}$ function

The first main function is *gpregression.m*. This function calculates the mean and variance from the input variables x_*, x_t based on the algorithm described in the theoretical section. As seen in Listing. 2.1 instead of an inverse matrix, a Cholesky decomposition is used here which improves the efficiency of the process [19]. The kernel matrix of the test data is one of the input variables here. This matrix is the largest and does not change for most of the runs of the algorithm so it is advisable to store it in the input and not recalculate it every time the mean and variance are calculated. The outputs are mean and variance which are heavily discussed in the theory section.

Listing 2.1: The *gpregression.m* code listing.

```
function [mean , variance ] = gpregression ( x_test , ...
  x_train, K_te , f_train ,l , sigma , kernel )

  x_train = transpose ( x_train );
  x_test = transpose ( x_test );

  K_tr = kernel ( euclideanDistanceBetweenTwo ...
    Sets ( x_train , x_train ) , sigma ,l );
  L = chol ( nearestSPD ( K_tr ));
  K_trte = kernel ( euclideanDistanceBetween ...
  TwoSets ( x_train , x_test ) , sigma , l );
  alpha = linsolve (L , linsolve ( transpose ( L ) , ...
    transpose( f_train )));
  mean = transpose ( transpose ( K_trte )* alpha );

  v = linsolve ( transpose ( L ) , K_trte );
  V = K_te - transpose ( v )* v ;
  variance = transpose ( diag (V ,0));
end
```

2.3 Automation of the $\mathcal{GPR}$

The previous section discusses only a single calculation of mean and variance function. To effectively solve problems, the program must be fully automated in case of choosing a new x_{t}. The GaussianProcess.m and *GPO.m* functions take care of the automation. The GaussianProcess.m function provides the repetition of the mean and variance calculation for the maximum number of $nMax$ iterations. Another situation to stop the program is when the minimum of $f_{\mathrm{t}}(x_{\mathrm{t}})$ is equal to or lower than the parameters goal which is the user defined value that sets the optimization goal. The function also has parameters *sis*, *rsis* and *sbom*. The sis is a number of samples based on variance each iteration, the *rsis* is a number of random samples every iteration, and *sbom* declares how often the sample based on the minimum of the mean function will be added. The example of setting up variables is shown in Listing. 2.2.

The input variables for the *gpregression.m* are defined in *GPO.m*. The *GPO.m* contains a loop that is repeating whole optimization more times and saving results

Listing 2.2: The *gpregression.m* code listing.

```
parameters . sigma = 1;
parameters . nMax = number_of_iterations ;
parameters . goal = -10;
parameters . covariancefunction = @expquadkernel ;

data . lim = x_limits ;
data . sis = number_of_samples_in_step ;
data . rsis = number_of_random_samples_in_step ;
data . sbom = sample_based_on_mean ;
```

of individual optimizations. This is because the gpregression.m contains random values, so for the testing it is important to run the algorithm more times for an objective review of the method.

2.4 Testing Functions

The method is tested on four different multivariate optimization testing functions for two up to five variables. The functions are Ackley function, Rastrigin's function, Sphere function, and Dixon and Price function [21],[22].

The Ackley's function for $x_i \in \langle$-32.768 32.768$\rangle$ with minimum $f(x) = 0$ at $x_i = 0$ [21]:

$$\mathbf{f}(\mathbf{x}) = -20 \cdot exp\left(-0.2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + a + exp(1) \tag{2.1}$$

The Rastrigin's function for $x_i \in \langle$-5.12 5.12$\rangle$ with minimum $f(x) = 0$ at $x_i = 0$ [21]:

$$\mathbf{f}(\mathbf{x}) = 10n + \sum_{i=1}^{n}[x_i^2 - 10cos(2\pi x_i))] \tag{2.2}$$

The sphere function for $x_i \in \langle$-5.12 5.12$\rangle$ with minimum $f(x) = 0$ at $x_i = 0$ [21]:

$$\mathbf{f}(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \tag{2.3}$$

The Dixon and Price function for $x_i \in \langle$-10 10$\rangle$ with minimum $f(x) = 0$ at $x_i = 2^{-\frac{2^i-2}{2^i}}$ [22]:

$$\mathbf{f}(\mathbf{x}) = (x_1 - 1)^2 + \sum_{i=1}^{n} i(2x_i^2 - x_{i-1})^2 \tag{2.4}$$

2.5 Testing $\mathcal{GPR}$ Optimization

When running a real optimization algorithm, it is necessary to choose the correct parameters with regard to the expected results. The first important parameters that need to be chosen are σ and l. For this application it is not wise to choose too low l as a low value combined with samples far apart could lead to a drop in μ and a constant $\mathbb{V}$ as shown in Fig. 1.3. The very high value can lead to problems too, particularly with rounding. The value $l = 15$ was chosen to avoid these drops of the minimum to zero.

Another important part is the choice of the number of iterations and the number of samples. In general, the more samples the more portable the method is. The same applies to the size of the matrix $\mathbf{K}(x_*,x_*)$. Again, the size cannot be too large. Too large matrix $\mathbf{K}(x_*,x_*)$ can cause a significant speed degradation. Since $\mathcal{GPR}$ is a stochastic process, it should be run several times to determine the average efficiency of the method. The number of runs was chosen as 100. Depending on the number of test functions and the number of runs for each function, the maximum test data size was 3000 and the number of samples was 1550 for optimal speed on the available hardware.

The number of iteration will be 500, the number of new points based on the variance in each iteration 2, the number of random steps in each iteration 1, and every 10 iterations will be a new point obtained from the minimum of the μ.

The Fig. 2.1 is a set of boxplots created from fitness errors obtained from runs. As previously mentioned this program is stochastic which causes the deviation of values. Since the same setting is used for problems of different dimensions, the fitness error increases significantly for more variables as shown in Fig. 2.1.

It is necessary to take into consideration that the program has been set up for a hundred runs and optimization of many problems to see the behavior for different problems. If used to optimize a single problem, more performance and time would be invested in it and the result would be much more accurate.

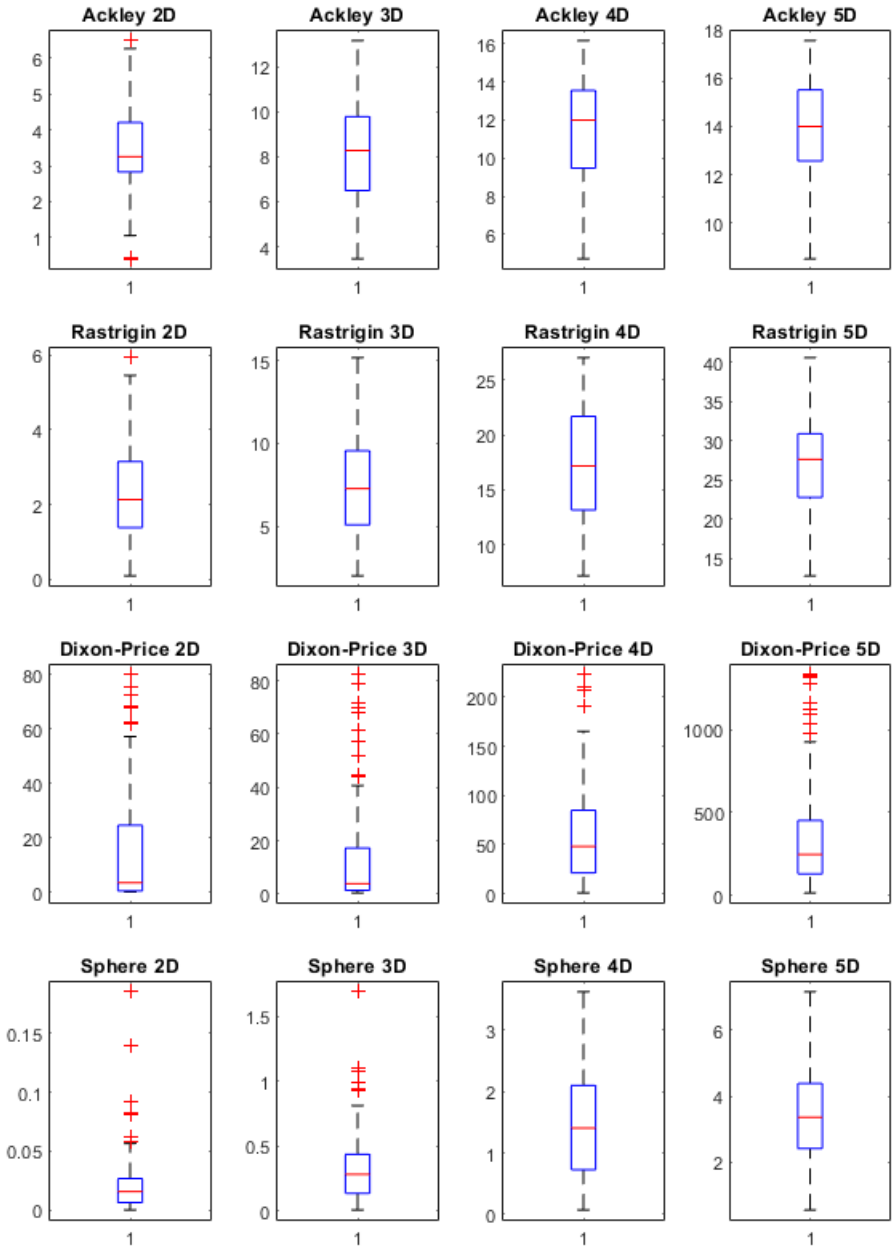

Fig. 2.1: Results of $\mathcal{GPR}$ optimizations.

2.6 Comparison with Other Stochastic Methods

Another important part is to compare the efficiency of the algorithm with other stochastics methods.. In this case, the Genetic algorithm GA, the Particle swarm optimization PSO, Algorithm of differential evolution DEA, and the Self-organizing migration algorithm $SOMA$ will be compared to the Gaussian process [17]. A MATLAB toolbox FOPS was used to test stochastic optimization methods [17]. To ensure an objective comparison of methods, all optimization algorithms in FOPS were simulated with a number of iterations of 100 and an agent number of 15 [17]. This reduction in agents ensures that the number of obejctive function computations to the algorithm is comparable to the number of Gaussian processes samples. The resulting number of samples is 1500 for FOPS and 1550 for Gaussian processes [17]. But the difference of 50 is negligible due to the differences in methods.

Again, the same settings as mentioned in the previous section were used. The number of runs is one hundred. The optimization algorithms were tested on the same optimization testing functions as $\mathcal{GP}$ optimizer. And the mean, minimum value, maximum value, and standard deviation were calculated from the results of these runs.

As can be seen in Tab. 2.1 , the accuracy of finding the optimum of Ackley´s function decreases as the number of dimensions increases [21]. The value of standard deviation is lower than the GA and the $SOMA$. With the increasing number of dimensions, the PSO algorithm's standard deviation increases much faster than the $\mathcal{GP}$ algorithm's.

Tab. 2.2 shows that for the Dixon-Price function the GP algorithm does not perform very well. Although for three variables it is quite close to the GA. And for all but the 2Dl problem, it comes close to the $SOMA$. In this case, the main problem is with high standard deviation.

Another test function is the Rastrigin function Tab. 2.3 [21]. This is a flat function containing many local extremes. Therefore even the $\mathcal{GP}$ algorithm has a low standard deviation, it is hard to find the minimum for multiple variables.

The last testing function is Sphere [21]. Due to the simplicity of the function, there is no problem for $\mathcal{GP}$ to approach the optimum. As for the twodimensional problem, the GA, the DEA, and the PSO all converge to 0 fairly quickly. But this is not the case for more variables, where $\mathcal{GP}$ achieve better results as displayed in Tab. 2.4.

Tab. 2.1: Results of Ackley function optimization

D	Algorithm	mean	minimum	maximum	standard deviation
2	$\mathcal{GPR}$	3.43	0.41	6.49	1.17
	GA	4.23	3.65e-4	10.12	2.84
	DEA	5.17e-7	2.19e-9	3.14e-6	6.12e-7
	PSO	5.54e-6	6.74e-8	2.03e-4	2.07e-5
	$SOMA$	1.39	4.44e-15	8.69	1.96
3	$\mathcal{GPR}$	8.05	3.46	13.16	2.28
	GA	6.30	1.31	14.69	3.28
	DEA	2.63e-4	7.36e-06	2e-3	2.90e-04
	PSO	0.418	1.49e-06	4.34	0.88
	$SOMA$	4.049	2.79e-04	14.57	2.99
4	$\mathcal{GPR}$	11.56	4.71	16.17	2.44
	GA	6.72	0.413	13.72	3.079
	DEA	22e-3	5.24e-4	1.84	0.184
	PSO	1.84	1.06e-4	7.46	1.39
	$SOMA$	5.32	0.122	13.24	2.92
5	$\mathcal{GPR}$	13.91	8.50	17.55	2.01
	GA	7.78	0.766	14.4	3.2
	DEA	99.8e-3	4.2e-3	8.37	0.84
	PSO	3.6	59e-3	12.21	2.2
	$SOMA$	7.43	1.53	14.03	3.079

Tab. 2.2: Results of Dixon and Price function optimization

D	Algorithm	mean	minimum	maximum	standard deviation
2	$\mathcal{GPR}$	15.33	19e-4	79.92	21.45
	GA	1.88	1.81e-4	16.02	3.99
	DEA	1.79e-4	4.66e-8	4e-3	5.19e-4
	PSO	7.47e-6	1.52e-17	17.87	7.47e-5
	$SOMA$	0.404	3.49e-30	7.47e-4	2.016
3	$\mathcal{GPR}$	13.65	0.42	82.32	19.57
	GA	6.586	5.5e-04	88.19	15.74
	DEA	13.5e-3	2.44e-4	0.115	0.02
	PSO	36.8e-3	8.18e-13	0.74	0.14
	$SOMA$	9.189	2.87e-6	139.8	18.92
4	$\mathcal{GPR}$	60.869	0.62	223.11	51.10
	GA	13.48	4.4e-3	106.9	28.1
	DEA	0.119	7.3e-3	0.724	0.11
	PSO	0.848	3.19e-8	19.42	2.28
	$SOMA$	24.29	4.9e-3	480	60.91
5	$\mathcal{GPR}$	352.77	13.51	1.33e+3	335.06
	GA	11.72	0.271	109.24	25.8
	DEA	0.49	56.1 e-3	2.40	0.304
	PSO	2.74	2.6e-3	26.78	5.3
	$SOMA$	129.98	0.82	1.87e+3	316.51

Tab. 2.3: Results of Rasrtigins function optimization

D	Algorithm	mean	minimum	maximum	standard deviation
2	$\mathcal{GPR}$	2.24	99e-3	5.93	1.21
	GA	1.65	2.4e-09	8.9	1.59
	DEA	0.04	0	0.995	0.196
	PSO	0.49	0	1.99	0.56
	$SOMA$	1.037	4.2e-12	4.98	1.075
3	$\mathcal{GPR}$	7.51	1.99	15.19	3.02
	GA	2.82	3.55e-9	10.14	2.21
	DEA	39e-3	1.82e-9	0.995	0.196
	PSO	1.56	2.47e-11	6.46	1.11
	$SOMA$	2.24	2.26e-4	6.09	1.59
4	$\mathcal{GPR}$	17.37	7.18	27.01	5.09
	GA	4.90	4.73e-9	12.95	2.82
	DEA	0.83	3.68e-5	1.07	0.258
	PSO	2.88	1e-3	9.7	2.016
	$SOMA$	3.75	41.8e-3	13.00	2.30
5	$\mathcal{GPR}$	27.03	12.79	40.57	6.05
	GA	7.03	0.995	15.16	3.35
	DEA	0.586	3.5e-3	3.19	0.671
	PSO	5.37	18.1e-3	19.10	3.62
	$SOMA$	5.61	1.13	14.43	3.15

Tab. 2.4: Results of Sphere function optimization

D	Algorithm	mean	minimum	maximum	standard deviation
2	$\mathcal{GPR}$	22.4e-3	1e-4	185e-3	27.2e-3
	GA	0	0	0	0
	DEA	0	0	0	0
	PSO	0	0	0	0
	$SOMA$	1.83	1.32e-8	13.99	3.082
3	$\mathcal{GPR}$	0.3387	9.8e-3	1.69	0.280
	GA	9.1e-13	0	9.1e-11	9.1e-12
	DEA	0	0	0	0
	PSO	0	0	0	0
	$SOMA$	6.62	4.4e-3	34.6	7.50
4	$\mathcal{GPR}$	1.47	72.8e-3	3.63	0.879
	GA	2.44e-4	0	24.4e-3	2.4e-3
	DEA	0.138	0	13.8	1.38
	PSO	0	0	0	0
	$SOMA$	13.8	27.2e-3	41.4323	10.44
5	$\mathcal{GPR}$	3.41	0.553	7.16	1.448
	GA	7.03	0.995	15.2	3.35
	DEA	0.586	3.5e-3	3.19	0.67
	PSO	5.37	18.1e-3	19.10	3.62
	$SOMA$	5.61	1.135	14.522	3.16

2.7 Improving $\mathcal{GPR}$ Performance Using DEA

2.7.1 DEA implementation

The optimization test results show that the optimizer converges too slowly to the solution, especially near the optimum limit of the fitness error function. This could be a problem when optimizing the antenna, due to the required accuracy of the parameter settings. The slow convergence to the solution is due to the randomness of the $\boldsymbol{x}_*$ dataset choice. Dataset $\boldsymbol{x}_*$ are generated randomly in the optimization limit region and for the best values its real fitness error is computed. This approach reduces the chance that dataset $\boldsymbol{x}_*$ will be at the location of the minimum of the $\mathcal{GP}$ model $\boldsymbol{\mu}$ and leads to slow convergence to the solution.

For this reason, the $\mathcal{GP}$ optimizer was extended with a differential evolution algorithm that efficiently finds the minimum of $\boldsymbol{\mu}$ in each iteration. The datasets $\boldsymbol{x}_*$ for $\mathbb{V}$ are still generated randomly and the ones with the largest value are evulated. This is primarily due to the possibility of finding solutions beyond known local minima. The differential evolution algorithm was chosen based on previous testing due to the best results.

Code bellow Listing 2.3 shows the function of the optimizer. Each iteration will first generate 5 agents of maximum $\mathbb{V}$ for random datasets and add them to the training values. In the next step, the trained $\mathcal{GP}$ model is taken and the minimum of $\boldsymbol{\mu}$ is found using differential evolution algorithm. Theí found minimum is added to the training dataset and in the next step the training dataset is simulated. These new values are then used to train the $\mathcal{GP}$, making the $\mathcal{GP}$ model more accurate each iteration.

Listing 2.3: Main optimization loop of $\mathcal{GP}$ including DEA.

```
x_train = generateRandomMatrix(x_limits,100);
    f_train = problem(x_train);
    min(f_train)
    l = 20;
    sigma = 2;
    for i = 1:50

        new_train = zeros(size(x_limits,1),1);

        for n=1:5
            x_test = generateRandomMatrix(x_limits,200);
            [mean,variance] = gpregression(x_test,...
                     x_train,f_train,l,sigma);
            [~,min_mean_pos] = min(mean);
            [~,max_variance_pos] = max(variance);

            new_train(:,n) = x_test(:,max_variance_pos);

        end
        [xDEA, fDEA] = DEA(@gpregression,x_limits,...
                      x_train,f_train,l,sigma)
        new_train = [new_train, xDEA];

        x_train = [x_train,new_train];
        f_train = [f_train,problem(new_train)];
        i
        min(f_train)
    end
```

2.7.2 Comparison of $\mathcal{GPR}$-DEA with $\mathcal{GPR}$

The $\mathcal{GPR}$ improved with DEA was tested on the same test functions as the $\mathcal{GPR}$. Each optimization was run 100 times and the mean, minimum, maximum and standard deviation were calculated from these results Tab. 2.5. Since the algorithm is not very suitable for fine optimization under large constraints, the main goal was to reduce the required computational power by reducing required nuber of samples.

The number of samples for which a real value was computed and against which the optimizer was tested was reduced from 1550 to 400. Despite this significant reduction, the results are comparable, in some cases better Fig. 2.2. This improvement will be advantageous in antenna optimization where the number of simulations needs to be reduced to a minimum. Due the relatively small difference between the results for different dimensions, it suggests that the algorithm works better for multidimensional problems Tab. 2.5.

Another improvement is the reduction of computational power required each iteration. This is mainly due to finding the minimum of the μ of $\mathcal{GP}$ model using DE. The new extended algorithm have roughly 10 times less iteration time. The speed of each iteration can also be improved by reduction of DE iterations or the population size of DE.

Tab. 2.5: Results of $\mathcal{GPR}$ optimizer with implemented DEA.

Testing function	D	mean	minimum	maximum	standard deviation
Ackley function	2	0.966	1.7e-2	2.910	0.795
	3	5.1026	1.9513	8.9450	1.4178
	4	5.9702	2.9853	14.4845	1.6400
	5	6.8240	3.3223	10.1520	1.2375
Dixon and Price function	2	78.623	2.789	318.784	57.266
	3	184.0865	49.5873	325.1861	60.3429
	4	285.1620	85.6652	445.5437	73.6399
	5	348.0343	182.6786	598.1808	90.9853
Rastrigin function	2	1.505	15.95	4.596	0.9274
	3	8.4151	0.2887	19.7780	4.0790
	4	25.5659	10.2964	41.2869	6.4602
	5	39.1326	17.0654	55.0604	7.4437
Spehre function	2	3.3224e-08	8.1099e-11	1.6710e-07	3.4399e-08
	3	1.2834e-05	7.1627e-07	4.9835e-05	9.5837e-06
	4	5.5075e-04	6.8326e-05	0.0015	2.9355e-04
	5	5.2e-3	5.5582e-04	1.09-2	2.394e-3

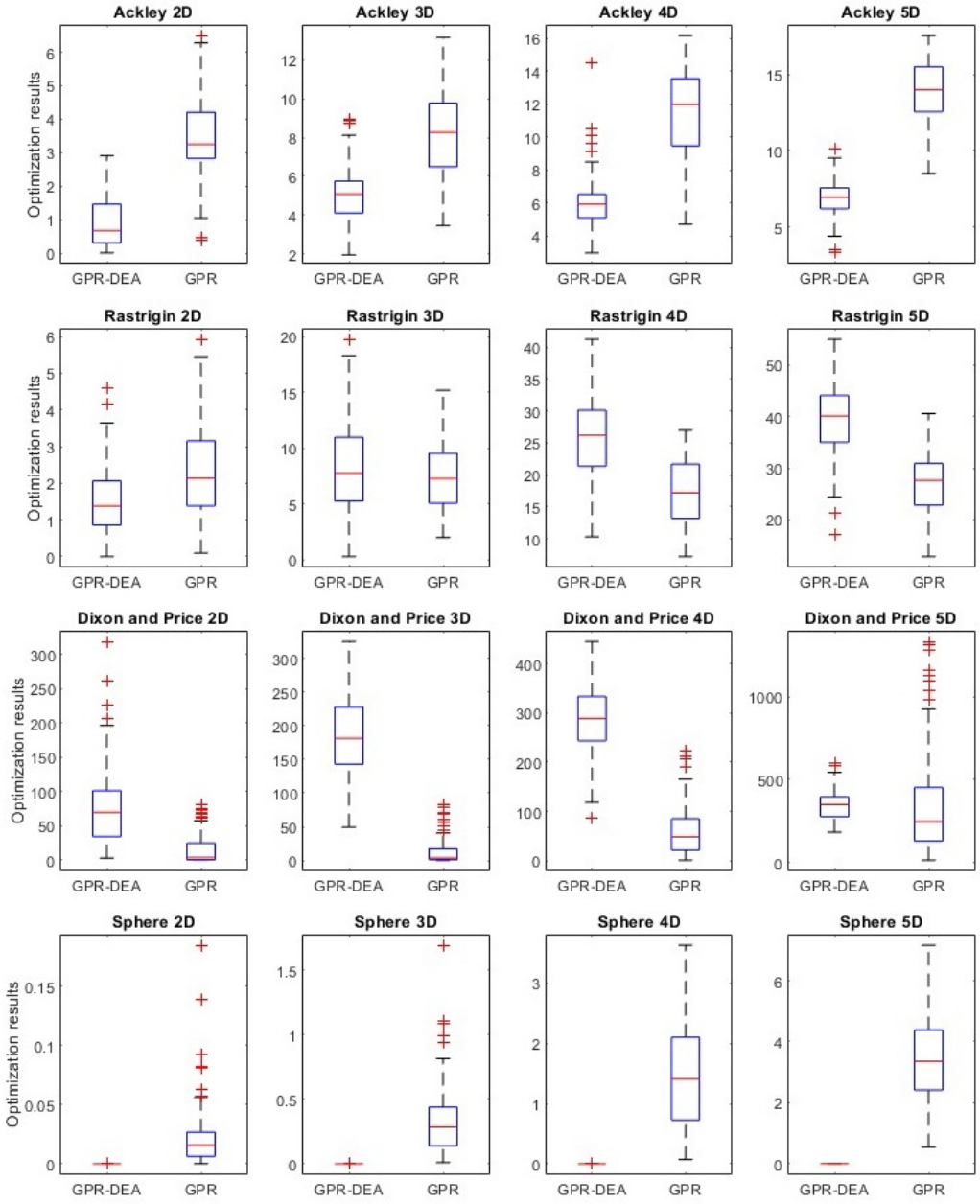

Fig. 2.2: Results of extended $\mathcal{GPR}$.

3 Antenna Design and Optimization

3.1 Prerequisite Antenna Design

The first step is to design the antenna on which the $\mathcal{GPR}$ optimizer will be tested. As already mentioned in the theoretical part, the correct choice of antenna type is essential to achieve the desired results. In this case, a rectangular patch antenna has been chosen, due to its simple design and small number of dimensions to optimise. The SMA connector was chosen as feed due to its availability, and the possibility of optimization through the position of the feed. This method of feeding has lower bandwidth than a proximity-coupled feed, but large bandwidth is not necessary for the 5.8GHz ISM band.

When designing it is necessary to take into consideration that with a lower permittivity the dimensions of the antenna increase. The substrate used is Arlon CuClad 217 with relative permittivity ε_{r}=2.17-2.2 and a height $h = 1.524$mm. The shape of the patch is chosen to be rectangular, without any cutouts or asymmetries. In this case, the circular polarization is invoked by the excentric position of the feed. The dimensions of the patch are calculated according to the equations [2]:

$$W = \frac{c}{2f_0\sqrt{\frac{\varepsilon_{\mathrm{r}}+1}{2}}} = 20,43mm \tag{3.1}$$

$$\varepsilon_{eff} = \frac{\varepsilon_{\mathrm{r}}+1}{2} + \frac{\varepsilon_{\mathrm{r}}-1}{2}\left[\frac{1}{\sqrt{1+12\left(\frac{h}{W}\right)}}\right] = 2,036 \tag{3.2}$$

$$L = \frac{c}{2f_0\sqrt{\varepsilon_{eff}}} - 0,824h\left(\frac{(\varepsilon_{eff}+0.3)\left(\frac{W}{h}+0.264\right)}{(\varepsilon_{eff}-0.258)\left(\frac{W}{h}+0,8\right)}\right) = 16.53mm \tag{3.3}$$

This calculation is for a linearly polarized antenna, with feed in the axis of the patch, so the antenna must be optimized for circular polarization. The optimization is done by changing the length L and width W of the patch and by changing the position of the feed in x-axis Px and y-axis Py.

The antenna is modelled and simulated by analysis software CST Studio Suite 2019. The optimization is done using the Nelder Mead Simplex Algorithm which is a feature of CST Studio Suite 2019 see Fig. 3.1 and Fig. 3.2. Dimensions and characetristics of optimized antenna are in Tab. 3.1 and Tab. 3.2. The antenna is $RHCP$ so it is not necessary to mirror the position of the feed. This procedure is to ensure that there is a suitable solution in the area where the GPR will be

tested. This prerequisite antenna will be used to determine the solution area and as a reference for the valid antenna parameters.

Tab. 3.1: Dimensions of optimized antenna.

W [mm]	L [mm]	Px [mm]	Py [mm]	Substrate width [mm]	Substrate length [mm]
15,079	16,128	5,425	4,665	60	60

Tab. 3.2: Parameters of optimized antenna.

f_0 [GHz]	$S11$ [dB]	AR [dB]	BW[MHz]	G[dBi]
5.807	-44.6	0.314	410	7.986

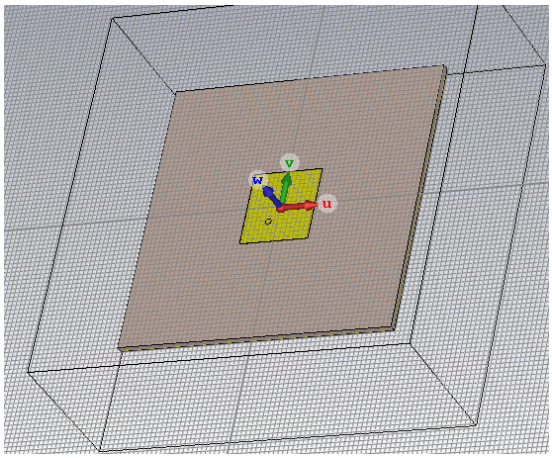

Fig. 3.1: Front view of the antenna in CST Studio Suite 2019.

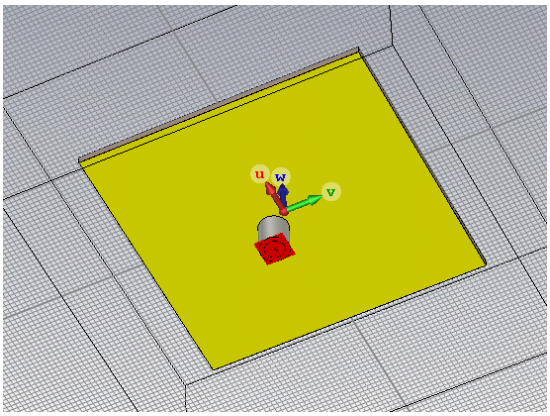

Fig. 3.2: Rear view of antenna in CST Studio Suite 2019.

3.2 GPR Antenna Optimization

3.2.1 Training Data Obtainment

As already mentioned CST Studio Suite 2019 was used to model the antenna. This program is also used to simulate the $S11$ and AR of the antenna to obtain training data. For this, control between CST and MATLAB needs to be implemented. CST Studio Suit is controlled by the simulateAntenna.m function Listing 3.1. This function initially sets the required parameters and then starts the solver. When the simulation is complete, the function exports the results to text files. The results in the text files are imported into a variable and the fitness error is calculated according to Eq:

$$f_{crit} = |S11_{goal} - S11_{sim}| \cdot w_1 + |AR_{goal} - AR_{sim}| \cdot w_2 \tag{3.4}$$

where w_1 is the weight of $S11$ and w_2 is the weight of AR. This value of the fitness error is further used for optimization.

3.2.2 Optimization

A GPR optimizer improved with DEA was used to optimize the circular polarized antenna. Fitness error is calculated by choosing $S11_{goal}$=50 and AR_{goal}=0. The weights are w_1=0.1 and w_2=0.5. The area of solution was $\pm$10% antenna soulution found in prewious part. The overall antenna optimization was divided into 3 sub-optimizations. This sub-optimization method was used due to the rounding in the GPR code and mainly in finding the nearest SPD matrix. This rounding causes inaccuracy, especially when fine-tuning over a large area.

The first part found the estimated location of the solution. The total number of simulated antennas in this step was 700. In this case, the best convergence to solve was in the first 200 samples. After the first 200 samples, the efficiency of the method dropped significantly Fig. 3.3. The antenna fitness error of the optimization is too high for the antenna to function properly Tab. 3.3. The limits for the next part were selected based on the best solution from this part.

The second part of the optimization is performed in the region of $\pm$2% of the solution found in the previous part. The number of simulations is also 700. In this part, the best convergence to the solution is in the first 600 simulations Fig. 3.3. The achieved solution already has a very low fitness error Tab. 3.3. The resulting antenna properties already match the desired solution.

A slightly different optimization method is used to fine tune the antenna. In this case, the new training data are no longer chosen based on $\mathbb{V}$ but only based on μ. This change is due to faster convergence to the solution at the cost of a lower chance

Listing 3.1: The *simulateAntenna.m* code listing.

```
function simulateAntenna(mws,W,L,Px,Py)
    invoke(mws, 'StoreParameter','W', W);
    invoke(mws, 'StoreParameter','L', L);
    invoke(mws, 'StoreParameter','Px', Px);
    invoke(mws, 'StoreParameter','Py', Py);
    invoke(mws, 'Rebuild');

    pause(0.5)
    solver = invoke(mws, 'Solver');
    invoke(solver, 'start');

    pause(0.5)
    invoke(mws,'SelectTreeItem',...
      '1D␣Results\S-Parameters\S2,2');
    ASCIIExport = invoke(mws,'ASCIIExport');
    invoke(ASCIIExport,'Reset');
    invoke(ASCIIExport,'SetVersion','2010');
    invoke(ASCIIExport,'FileName',append(...
       'C:\Users\xniede04\Documents\Optimize\',...
           's11','.txt'));
    invoke(ASCIIExport,'Execute');

    pause(0.5)
    invoke(mws,'SelectTreeItem','Tables\1D␣Results\AR_2');
    ASCIIExport = invoke(mws,'ASCIIExport');
    invoke(ASCIIExport,'Reset');
    invoke(ASCIIExport,'SetVersion','2010');
    invoke(ASCIIExport,'FileName',append(...
       'C:\Users\xniede04\Documents\Optimize\',...
            'AR','.txt'));
    invoke(ASCIIExport,'Execute');
end
```

of finding a different global minimum. This part of the optimization was performed in the region of ±2% of the best result of the previous part . The optimization took only 101 simulations to find a sufficient solution Tab. 3.3.

A total of 1501 solver runs were needed to optimize the antenna. The optimization is very computationally intensive, so it was not possible to run the optimization in sufficient numbers to determine the standard deviation of the result. The nature of the resulting optimization corresponds to testing on test functions. As on the test functions, the GP model has difficulty modeling narrow local minimums for large areas.

Tab. 3.3: Optimized dimensions of antenna.

	W	L	Px	Py	fitness error
first part of optimization	14.95	16.04	5.12	4.82	2.8291
second part of optimization	15.10	16.12	5.45	4.76	0.7976
fine optimization	15.11	16.14	5.34	4.79	0.0818

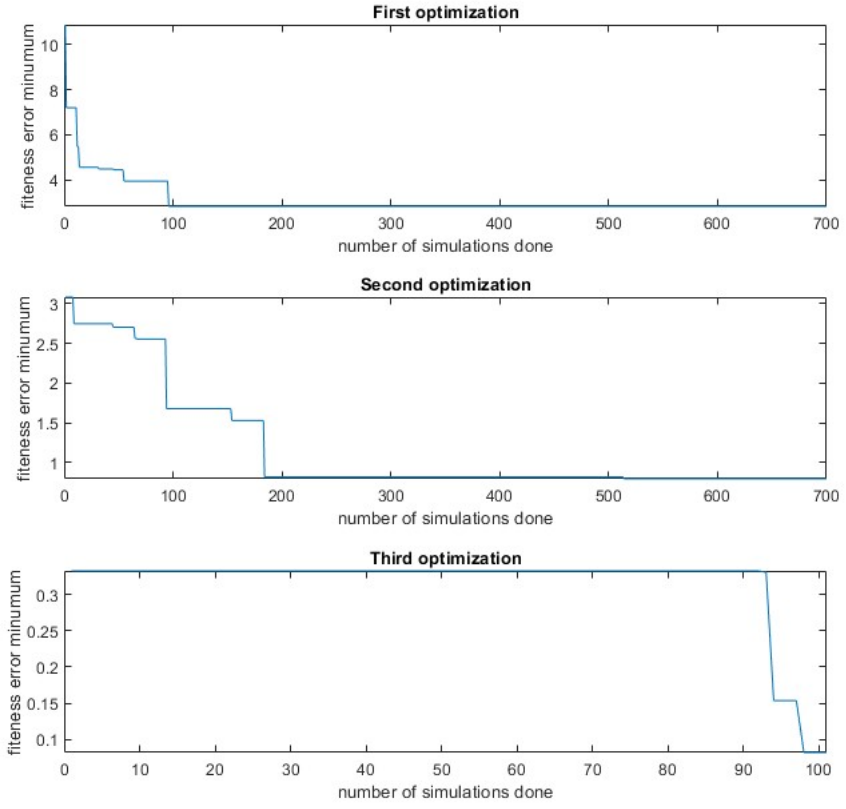

Fig. 3.3: Evolution of the minimum total fitness error during optimization.

3.2.3 Optimized Antenna Results

In the analysis of the results, the antenna designed using $\mathcal{GPR}$ is compared with the solution of the prerequisite antenna see Tab. 3.4. The antenna designed using $\mathcal{GPR}$ achieves the desired results. The parameters of this antenna are comparable to the targets determined by the prerequisite antenna. The $\mathcal{GPR}$ optimized antenna achieves generally better parameters. These differences in $S11$, f_0, G, and BW are negligible, considering the errors that may occur during fabrication. The improvement in the AR parameter from 0.314dB to 0.160dB is a significant, because of its importance. Radiation patterns are almost same see Fig. 3.6. The designed antenna meets the application requirements, the frequency and bandwidth are suitable for the 5.8GHz ISM band. The optimization results are adequate for antenna fabrication.

Tab. 3.4: Results of optimized antennas.

Antenna	f_0 [GHz]	$S11$ [dB]	AR [dB]	BW[MHz]	G[dBi]
Prerequisite antenna	5.807	-44.6	0.314	410	7.986
$\mathcal{GPR}$ optimized antenna	5.801	-46.0	0.160	405	7.987

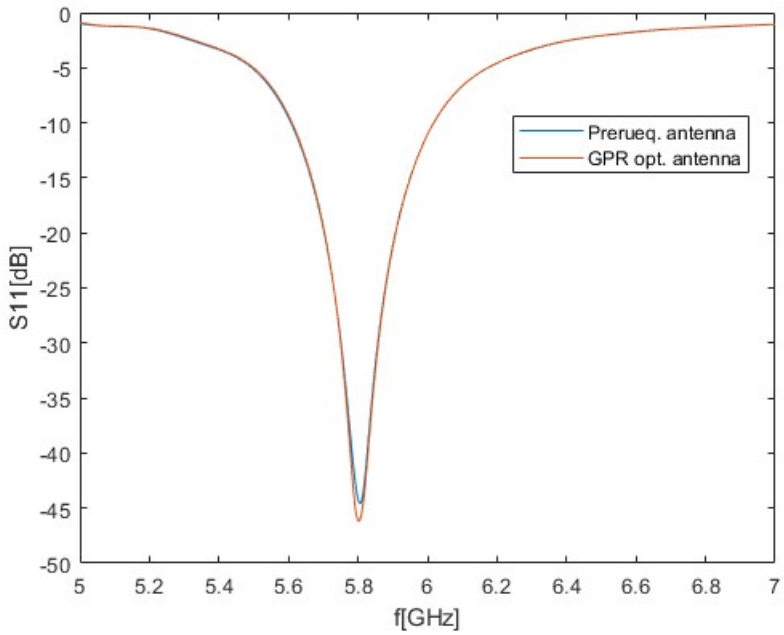

Fig. 3.4: $S11$ parameter of antenna prerequisite and $\mathcal{GPR}$ optimized antenna.

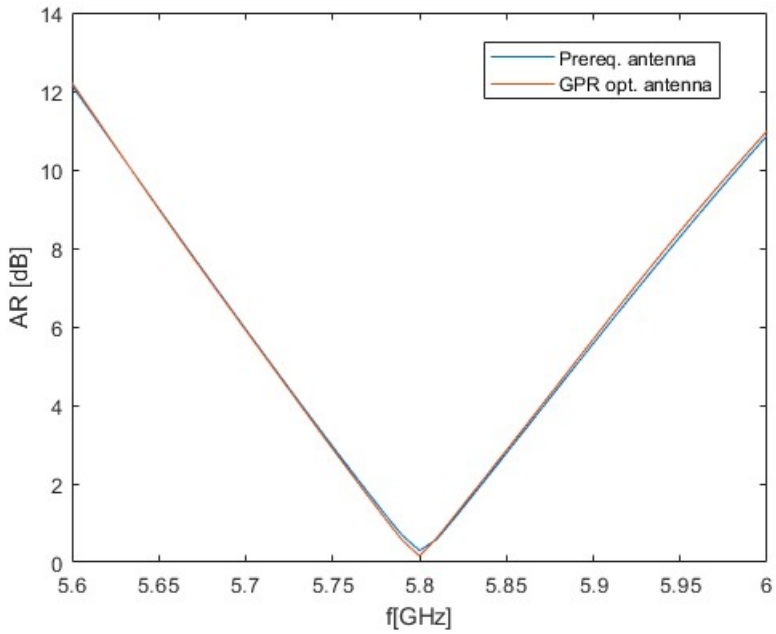

Fig. 3.5: AR of antenna prerequisite and $\mathcal{GPR}$ optimized antenna.

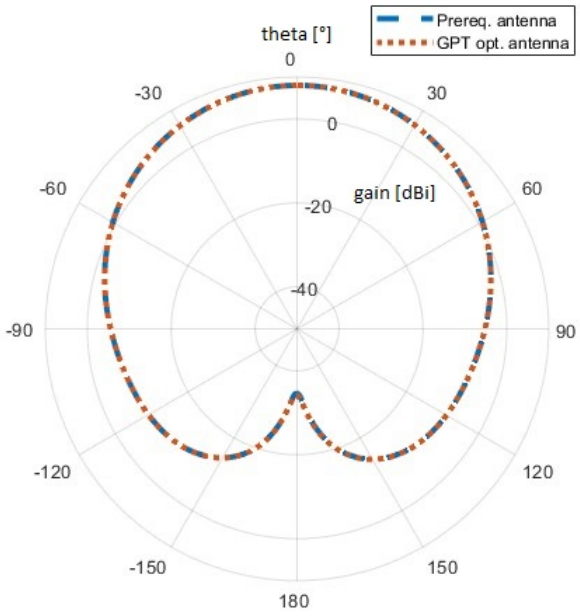

Fig. 3.6: RHCP radiation pattern of antenna prerequisite and $\mathcal{GPR}$ optimized antenna.

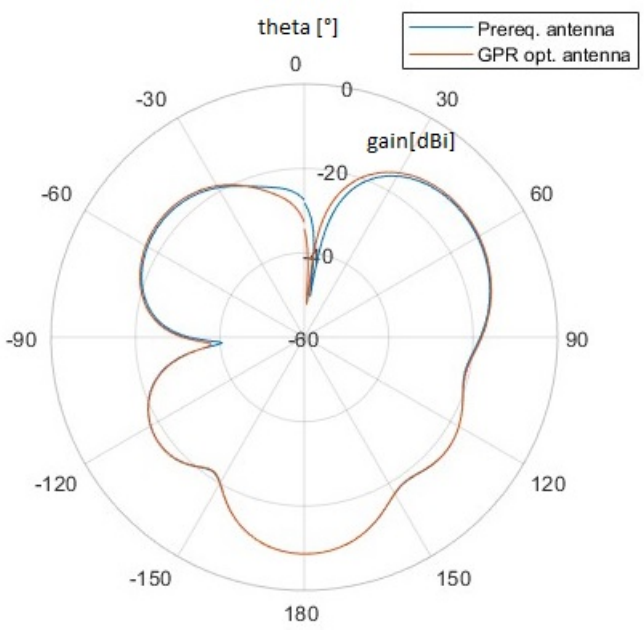

Fig. 3.7: LHCP radiation pattern of antenna prerequisite and $\mathcal{GPR}$ optimized antenna.

4 Realization of the Antenna

4.1 Antenna Fabrication

The antenna is made on the PTFE substrate CuClad217 h=1.524 mm. The front and back side is coated with copper. The front side of the substrate is etched in the shape of a patch Fig. 4.1. On the rear side, a circular shape is etched to isolate the SMA connector feed Fig. 4.2. A 1.3 mm diameter hole is drilled for the feed. The SMA connector is trimmed to the desired length and fixed in place by soldering to the ground Fig. 4.4. From the front side, the feed is soldered to the patch Fig. 4.3.

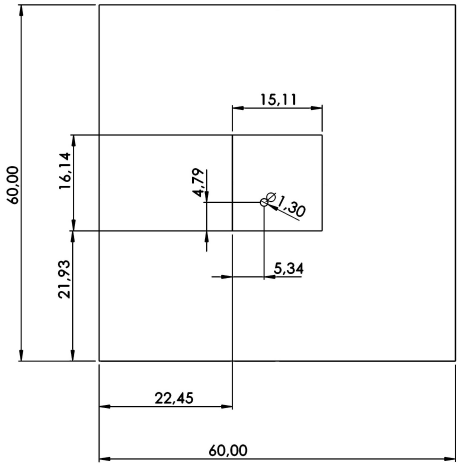

Fig. 4.1: Front view of antenna dimensions.

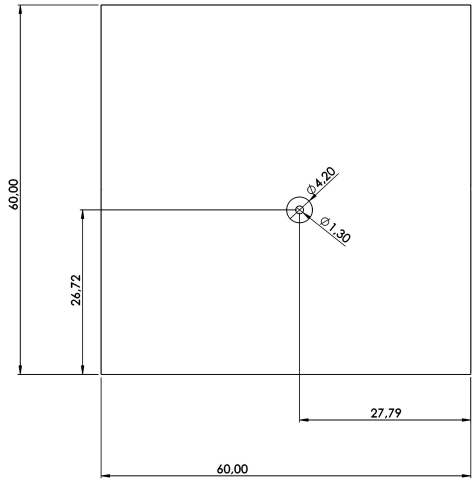

Fig. 4.2: Rear view of antenna dimensions.

Fig. 4.3: Front photo of antenna.

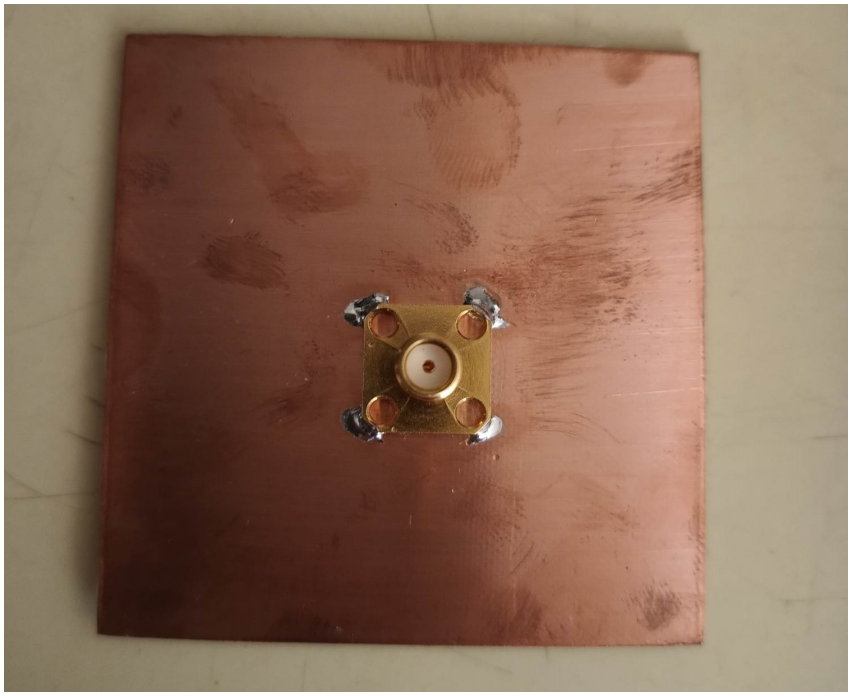

Fig. 4.4: Rear photo of antenna.

4.2 Comparison of Measurement with Simulation

Measurements of $S11$ parameter, radiation characteristics, axial ratio, and gain were done for the fabricated antenna. Measurement of the S11 parameter was done on a vector circuit analyzer. The rest of the measurements were done in a fully anechoic room. The measurements revealed a deviation in the resonant frequency of 2.34%, resulting in an absolute shift from 5.8 GHz to 5.936 GHz Tab. 4.1. The reflection coefficient at this frequency is -19.17 dB. Additionally, there is a degradation in the bandwidth $BW = 346$ MHz. The measured antenna also have a slightly higher axial ratio $AR = 0.315$ dB see Fig. 4.6. Considering its value and the fact that its minimum occurs at a resonant frequency of the antenna, this AR can be considered as a good result. The radiation characteristics align with the simulations and indicate successful achievement of right-hand circular polarization $RHCP$. The half-power beam width in elevation $HPBWE$ was measured to be 79°. The antenna gain is 7.5 dBi at 5.93 GHz and 6.85 dBi at 5.8 GHz.

Tab. 4.1: Antenna simulation and measurement results.

Antenna	f_0 [GHz]	$S11$ [dB]	AR [dB]	BW [MHz]	G [dBi]
simulation ε_r=2.2	5.801	-46.0	0.160	405	7.987
simulation ε_r=2.1	5.920	-46.18	0.213	424	8.088
measured antenna	5.936	-19.17	0.315	346	7.5

The difference between the simulated and measured resonant frequency may be slightly affected to manufacturing errors. In the case of a fabrication error, the $S11$ and AR would significantly degraded. The main influence may be the lower relative permittivity of the substrate used. When simulating the antenna with lower ε_r, similar results to the measured values were obtained Tab. 4.1. The antenna simulated with a relative permittivity of $\varepsilon_r = 2.1$ matches the measured antenna.

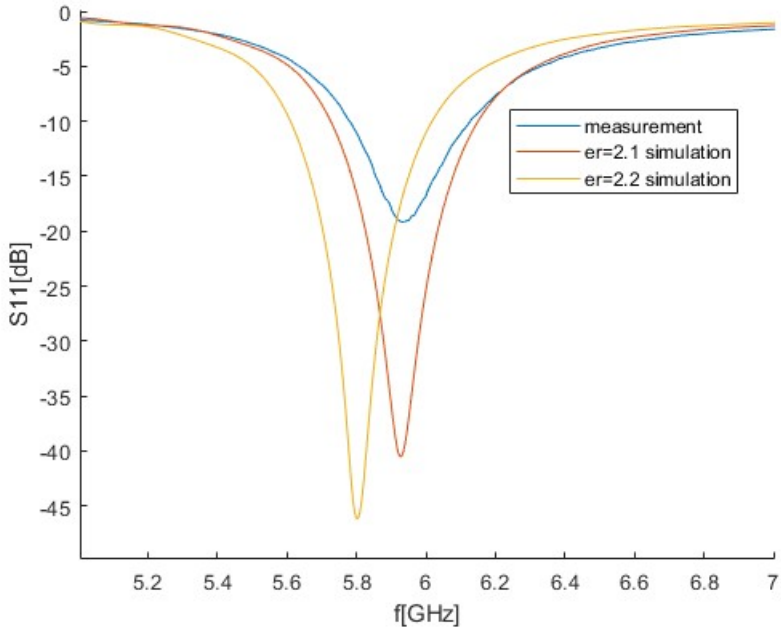

Fig. 4.5: $S11$ parameter of measured and simulated antennas.

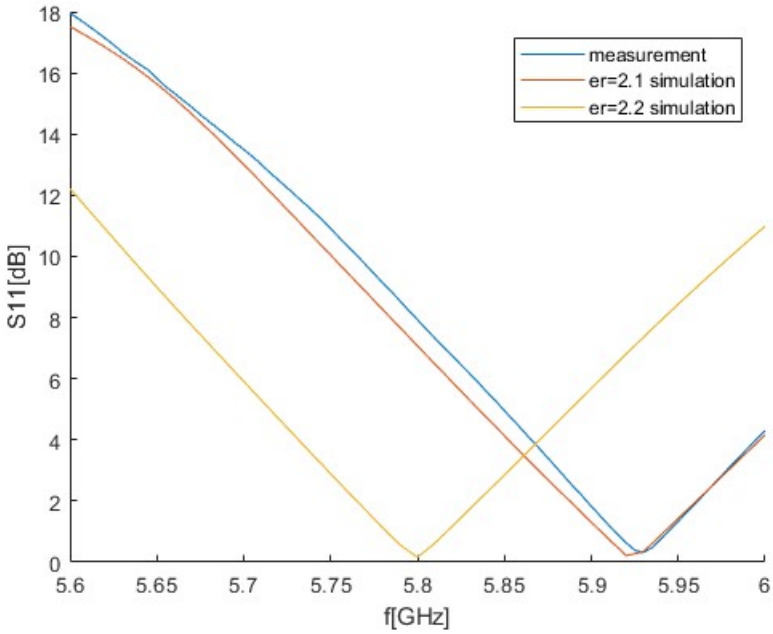

Fig. 4.6: AR of measured and simulated antennas.

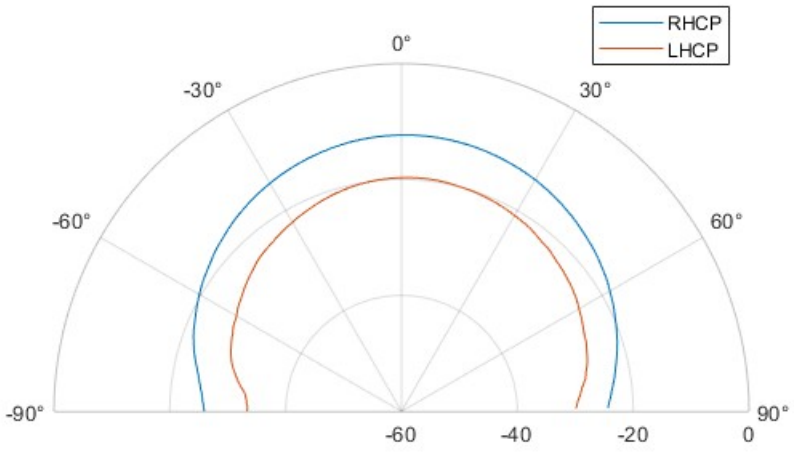

Fig. 4.7: Radiation patterns of antenna longitudinal plane for 5.8 GHz.

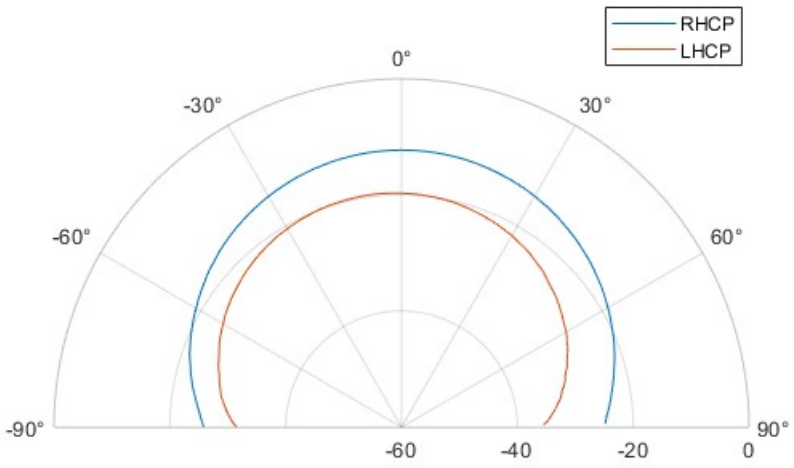

Fig. 4.8: Radiation patterns of antenna transverse plane for 5.8 GHz.

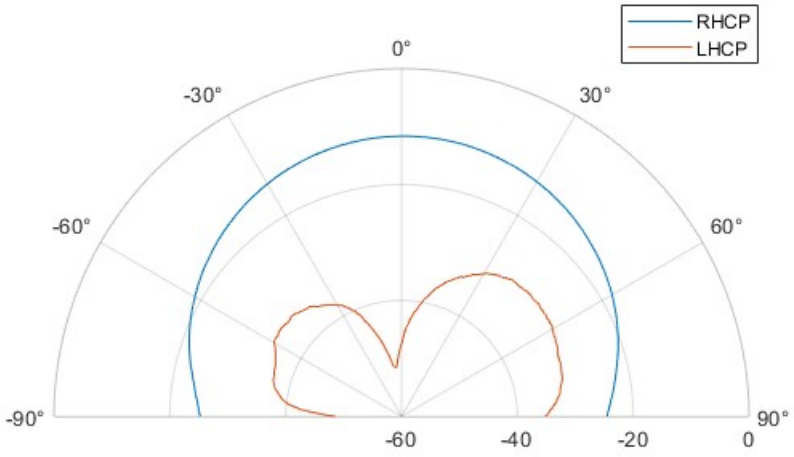

Fig. 4.9: Radiation patterns of antenna longitudinal plane for 5.93 GHz.

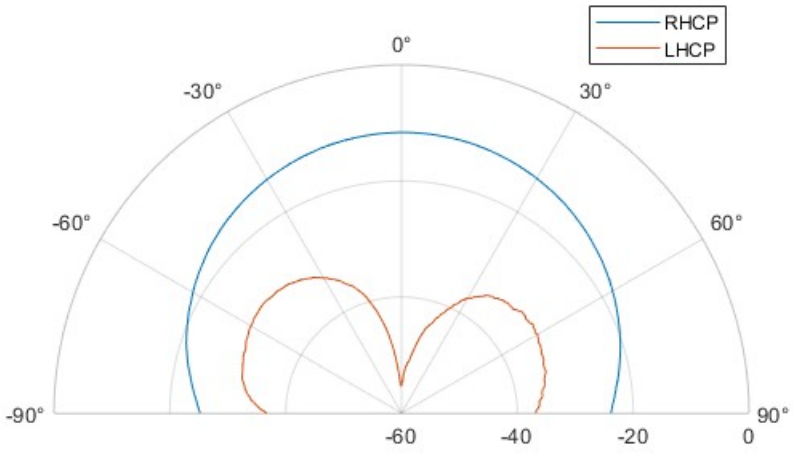

Fig. 4.10: Radiation patterns of antenna transverse plane for 5.93 GHz.

5 Conclusion

For the implementation of the optimization algorithm, a Gaussian process regression was chosen. For kernel matrix the squared exponential covariance function was chosen due to its smoothness and wide use. This metod was implemented in MATLAB.

Since this code was intended to be used as an optimization algorithm, it was necessary to extend it with a function for the automatic selection of additional points. This implementation evaluates the choice of a new training point based on the uncertainty in the function domain and the minimum of the mean function.

The algorithm was then tested on testing functions where it became clear that the size of the kernel matrix had to be taken into account. For large datasets and a large number of variables, the kernel matrix took on large dimensions and slowed down the algorithm significantly. The performance demand was many times higher than other algorithms such as DEA, PSO, GA and $SOMA$. In this state, the algorithm did not have many advantages over faster and better performing methods. One of the advantages is the possibility of keeping the $\mathcal{GPR}$ modeled problem.

The $\mathcal{GPR}$ optimization results was compared with DEA, PSO, GA and $SOMA$. The best results were achieved by the sphere function where it achieved higher accuracy than $SOMA$, PSO, and GA for multiple input variables. The worst results were achieved for the Dixon-Price function for more dimensions or more variables the method was inefficient.

Due to the poor efficiency results of the algorithm, it was extended with DEA. In each iteration, the algorithm generates random test data for which $\mathbb{V}$ and μ are calculated. From these data, it selects the appropriate ones for training. This approach results in a small probability that the test data will be at the point of local minimum. In order to efficiently find the μ minimum, the algorithm has been extended with DEA. The newly modified algorithm was tested on the same test functions. The number of samples was reduced from 1550 to 400 and despite this the optimizer achieved comparable and in some cases better results.

For antenna optimization, the algorithm extended with DEA was used. The designed antenna had 4 parameters and testing was done in the region of $\pm 10\%$ of the solution value. 1501 simulations were required to optimize the antenna. During the optimization, it was apparent that $\mathcal{GPR}$ has a fine optimization problem at large limits. In the case of choosing a smaller range of limits, the algorithm was also capable of fine tuning. The optimized antenna achieved the desired results Tab. 3.4. To better analyze the results of this optimization, the optimizer would need to be run more times. Unfortunately, this was not possible due to the computational time of the simulations.

The optimized antenna has been fabricated and measured. The only significant deterioration of the antenna is the shift of f_0 to 5.93 GHz. Due the good AR value at this frequency, the error can be caused by smaller ε_r value of used substrate compare to declared. This is supported by simulations where a substrate with ε_r=2.1 achieves similar results to the measurements.

In general, GPR is suitable for optimizations, mainly due to its trainability and the possibility to use the model later. The computational performance and the quality of the results are determined by the way the model is trained. Another factor is the choice of the covariance function, which is choice is based on the expected course of the problem. When using this algorithm, it is necessary to take into account the greater demand for computing power. The speed of iterations can be influenced by changing the algorithm parameters, but it is still slower than DEA, PSO, $SOMA$, or DE.

Bibliography

[1] WILLIAMS, Christopher KI; RASMUSSEN, Carl Edward. *Gaussian processes for machine learning.* Cambridge, MA: MIT press, 2006.

[2] MILLIGAN, Thomas A. *Modern antenna design.* New York: McGraw-Hill, 1985. ISBN 978-0471457763

[3] FORRESTER, A. I. J.; SOBESTER, A.; KEANE, A. J. *Engineering Design via Surrogate Modelling.*. A John Wiley and Sons Ltd Publication, Southampton, 2008. ISBN 9780470060681.

[4] LIU, Miao, et al. Gaussian processes for learning and control: A tutorial with examples.*IEEE Control Systems Magazine*, 2018, 38.5: 53-86.

[5] SÄRKKÄ, Simo; PICHÉ, Robert. On convergence and accuracy of state-space approximations of squared exponential covariance functions. In: 2014 *IEEE International Workshop on Machine Learning for Signal Processing (MLSP).* IEEE, 2014. p. 1-6.

[6] PETERSEN, Kaare Brandt, et al. *The matrix cookbook.* Technical University of Denmark, 2008, 7.15: 510.

[7] CHEN, Zexun; WANG, Bo. How priors of initial hyperparameters affect Gaussian process regression models. *Neurocomputing*, 2018, 275: 1702-1710.

[8] PRICE, Kenneth; STORN, Rainer M.; LAMPINEN, Jouni A. *Differential evolution: a practical approach to global optimization.* Springer Science & Business Media, 2006.

[9] CHAKRABORTY, Uday K. (ed.).*Advances in differential evolution.* Springer, 2008.

[10] BREST, Janez, et al. Performance comparison of self-adaptive and adaptive differential evolution algorithms.*Soft Computing*, 2007, 11.7: 617-629.

[11] KADLEC, Petr. Design of Artificial Neural Network for Antenna Synthesis using the Optimization with Variable Number of Dimensions. In: 2022 *32nd International Conference Radioelektronika (RADIOELEKTRONIKA).* IEEE, 2022. p. 01-06.

[12] PATEL, Shobhit K.; KOSTA, Y. P. E-shape microstrip patch antenna design for GPS application. In: 2011 *Nirma University International Conference on Engineering.* IEEE, 2011. p. 1-4.

[13] BALANIS, Constantine A. *Antenna theory: analysis and design*. 3rd ed. Hoboken: Wiley-Interscience, 2005. ISBN 978-0-471-66782-7.

[14] GAO, Steven Shichang; LUO, Qi; ZHU, Fuguo. *Circularly polarized antennas*. John Wiley & Sons, 2014.

[15] LEE, Kai Fong; LUK, Kwai Man; LAI, Hau Wah. *Microstrip patch antennas*. World Scientific, 2017.

[16] MISHRA, Ranjan. An overview of microstrip antenna. *HCTL Open International Journal of Technology Innovations and Research (IJTIR)*, 2016, 21.2: 39-55.

[17] MAREK, Martin; KADLEC, Petr; ČAPEK, Miloslav. FOPS: A new framework for the optimization with variable number of dimensions. *International Journal of RF and Microwave Computer-Aided Engineering*, 2020, 30.9: e22335.

[18] CAPEK, Miloslav, et al. AToM: A versatile MATLAB tool for antenna synthesis.*12th European Conference on Antennas and Propagation (EuCAP 2018)* 2018.

[19] KRISHNAMOORTHY, Aravindh; MENON, Deepak. Matrix inversion using Cholesky decomposition. In:*2013 signal processing: Algorithms, architectures, arrangements, and applications (SPA)*. IEEE, 2013. p. 70-72.

[20] John D'Errico, *nearestSPD*[online]. 2013 [cit. 21.12.2021]. Dostupné z: `https://uk.mathworks.com/matlabcentral/fileexchange/42885-nearestspd`

[21] MOLGA, Marcin; SMUTNICKI, Czesław. Test functions for optimization needs. *Test functions for optimization needs*, 2005, 101: 48.4.

[22] JAMIL, Momin; YANG, Xin-She. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 2013, 4.2: 150-194.

Symbols and abbreviations

$\mathcal{GP}$	Gaussian process
$\mathcal{GPR}$	Gaussian process regression
$\mathcal{N}$	Gaussian/normal distribution
$\mathbb{E}$	Expectation
$\mathbf{K}$	Kernel matrix
$\mathbf{k}$	Covariance function
σ	Vertical scale variable of covariance function
l	Horizontal scale variable of covariance function
$\boldsymbol{d}$	Euclidean distance between two points
x_*	Testing data inputs
μ	Mean of the Gaussian process regression
$\mathbb{V}$	Variance
x_t	Training data inputs
$f_\mathrm{t}(x_\mathrm{t})$	Training data outputs
ϵ	Pressumed noise
$\sigma^2 I$	Noise variable
DE	Difirential evolution
CR	Crossover Ratio
F	Scaling factor
ε_r	Relative permittivity
Γ	Reflection coefficient
$LHCP$	Left handed circular polarization
$RHCP$	Right handed circular polarization
GA	Genetic algorithm

DEA	Differential evolution algorithm
PSO	Particle swarm optimization algorithm
$SOMA$	Self-organizing migrating algorithm
L	Length of patch
W	Width of patch
Px	Position of feed in X axis
Py	Position of feed in Y axis
h	Height of substrate
AR	Axial ratio of polarization
$S11$	Reflection coefficient
$HPBWE$	half-power beam width in elevation

6 OBSAH ELEKTRONICKÉ PŘÍLOHY

```
\
|  navrh_anteny.jpg
|  navrh_anteny2.jpg
|  Tree.txt
|
+---GPDEA_testovani
|  \---GPDEA
|      |  2D.mat
|      |  3D.mat
|      |  4D.mat
|      |  5D.mat
|      |  Ackley.m
|      |  DEA.m
|      |  DixonPrice.m
|      |  euclideanDistanceBetweenTwoSets.m
|      |  expquadkernel.m
|      |  generateRandomMatrix.asv
|      |  generateRandomMatrix.m
|      |  gpregression.m
|      |  main.m
|      |  Rastrigin.m
|      |  Sphere.m
|      |
|      +---DEA
|      |      crossoverDE.m
|      |      DE.m
|      |      mutationDE.m
|      |      selectionDE.m
|      |
|      \---NearestSymmetricPositiveDefinite
|          |  nearestSPD.m
|          |  nearestSPD_demo.m
|          |
|          \---html
|                 nearestSPD_demo.html
|                 nearestSPD_demo.png
|                 nearestSPD_demo_01.png
|
+---GP_optimalizace_bez_DE
|  |  Ackley.m
|  |  Ackley2D.mat
|  |  Ackley3D.mat
|  |  Ackley4D.mat
|  |  Ackley5D.mat
|  |  Boxplots.m
|  |  DixonPrice.m
```

```
|   |   DixonPrice2D.mat
|   |   DixonPrice3D.mat
|   |   DixonPrice4D.mat
|   |   DixonPrice5D.mat
|   |   euclideanDistanceBetweenTwoSets.m
|   |   expquadkernel.m
|   |   GaussianProcess.m
|   |   GPO.m
|   |   gpregression.m
|   |   Rastrigin.m
|   |   Rastrigin2D.mat
|   |   Rastrigin3D.mat
|   |   Rastrigin4D.mat
|   |   Rastrigin5D.mat
|   |   Sphere.m
|   |   Sphere2D.mat
|   |   Sphere3D.mat
|   |   Sphere4D.mat
|   |   Sphere5D.mat
|   |
|   \---NearestSymmetricPositiveDefinite
|       |   nearestSPD.m
|       |   nearestSPD_demo.m
|       |
|       \---html
|               nearestSPD_demo.html
|               nearestSPD_demo.png
|               nearestSPD_demo_01.png
|
+---mereni_anteny
|       AR.txt
|       LHP.txt
|       PodelRov_LHCP.dat
|       PodelRov_RHCP.dat
|       PricRov_LHCP.dat
|       PricRov_RHCP.dat
|       RHP.txt
|       s11.txt
|
+---optimalizace_antenna
|   |   Ackley.m
|   |   antennaSimulation.m
|   |   AR.txt
|   |   CP_antenna.cst
|   |   DEA.m
|   |   euclideanDistanceBetweenTwoSets.m
|   |   expquadkernel.m
|   |   FirstOptimization.mat
```

```
|   |   generateRandomMatrix.m
|   |   gpregression.m
|   |   lastOPT.mat
|   |   main.m
|   |   makeCriterial.m
|   |   nextOpt.mat
|   |   s11.txt
|   |   simulateAntenna.m
|   |
|   +---DEA
|   |       crossoverDE.m
|   |       DE.m
|   |       mutationDE.m
|   |       selectionDE.m
|   |
|   \---NearestSymmetricPositiveDefinite
|       |   nearestSPD.m
|       |   nearestSPD_demo.m
|       |
|       \---html
|               nearestSPD_demo.html
|               nearestSPD_demo.png
|               nearestSPD_demo_01.png
|
\---vysledky_CST
    +---ER21
    |       AR.txt
    |       radiation.txt
    |       S.txt
    |
    +---GPROpt
    |       AR.txt
    |       radiation.txt
    |       S.txt
    |
    \---InitialCSTopt
            AR.txt
            Radiation.txt
            S11.txt
```