MODPY

Python Module

Technical Description

–

Python Module for Uncertainty Proxy Modelling

Author:
Frederik Lehn

July 29, 2021

Contents

Nomenclatue

- Proxy Model: An alternative representation of a model.
- Simulation: Expensive calculation of the forward problem (such as computational fluids dynamics)
- Simulator: Software used for performing a simulation with a given set of parameters
- Solution Space: A combination of parameters which is calibrated through an optimization process.
- Objective Function: A function driving the objective of an optimization process.

Table 1: Nomenclature

Variables	Description
ρ	Pearson's Correlation Coefficient
ρ_s	Spearman's Rank Correlation Coefficient
τ	Kendall's Rank Correlation Coefficient
$\mathbb{R}$	The set of real numbers
$\mathbb{R}_+$	The set of strictly positive reals
$\mathbb{R}_{\geq 0}$	The set of positive reals
$\mathbb{N}_+$	The set of strictly positive integers

Introduction

The purpose of `modpy` is to provide a comprehensive python package for black-box optimization of problems with computationally expensive function calls (simulations). The underlying approach utilized is a combination of proxy models and experimental designs to achieve maximum information with a minimum number of simulations. The package is designed to handle uncertainty on parameters all along the modelling chain. The philosophy is to navigate in the stochastic domain by characterizing the uncertainty space as opposed to converging towards a single deterministic solution. The cases typically benefiting from such a modelling approach are time-series problems with uncertain initial conditions. These cases often arise in engineering where physical laws are modelled which are stationary in time, but parameterized with uncertain and/or sparse data sets.

The package is specifically well positioned to handle cases following a given workflow:

1. Calibrate model parameters using measurements
2. Optimize design parameters given some engineering or financial objective
3. Perform forecasts to quantify the value of a project

Individual phases of the workflow can be ignored, but the full value of the approach is only achieved when all phases are combined. An example where the full workflow is common is the Oil & Gas industry. Here wells are added to a discretized geological model and the production is simulated in time. Such simulations usually take anywhere between 30 minutes and several days depending on the size and complexity of the model. A typical workflow is to *history match* the model to existing production data. Then optimize the placement of new wells and subsequently forecast the incremental production of the new development through simulation. A different example only using the design optimization is the Aerospace industry where various design parameters can be tuned to optimize the flow of air around an airfoil. Again the flow is evaluated using computational fluid dynamic simulations, which can be computationally very expensive. In both of these examples the *simulators* used to predict production from a well and flow around an airfoil respectively, is what is referred to as block-boxes.

The first part of this Technical Description outlines the three main phases of the workflow and how they are solved. The second part describes all the mathematical theory involved in performing the modelling done during the process. Lastly, a series of case studies are provided showing how it works in practice.

Part I

Modelling Philosophy

The modelling approach adopted by `modpy` suffers from the *Curse of Dimensionality*. This results in a feasibility limit on the number of uncertain or design parameters which can be used in the optimization processes. As a result any problem has to be parameterized in such a way that it can be modelled by a limited number of parameters. The difficulty of such a parameterization and other imposed limitations is case specific. Alternative methods exists for which the computational requirements are not proportional to the dimensionality of the problem. An example of this is the Ensemble Kalman Filter which can handle millions of parameters and the necessity of such an approach arises for instance in Weather Forecasting. Such approaches comes with other drawbacks, and typically they require a strong integration between the model building application, the simulator and the optimizer. This makes it difficult to handle such problems in a black-box setting.

1. Model Calibration

Model Calibration is the process of tuning unobserved parameters by matching simulated responses to data. This is an inverse problem and consequently a unique solution is not guaranteed. Further to this, measurement errors of instruments introduces uncertainty on the data and the parameterization of the problem introduces model errors. In plain terms this means using a simplified model to match some potentially spurious data with no guarantee of reaching the global optimum. For this reason the notion of finding the "true" parameters is erroneous - at best it is possible to narrow the solution to the set of *most likely* candidates. As most physical parameters behave continuously and not discretely, this of course is not a set of parameters, but rather a joint probability density function.

An inverse problem is solved by iteratively calculating the forward problem with a provided set of parameters, and measuring the calculated response relative to the observed data. Depending on the discrepancy between the response and the data the set of parameters are adjusted and the forward problem is calculated again. The discrepancy between the calculated response and the observed data is what is known as the objective function.

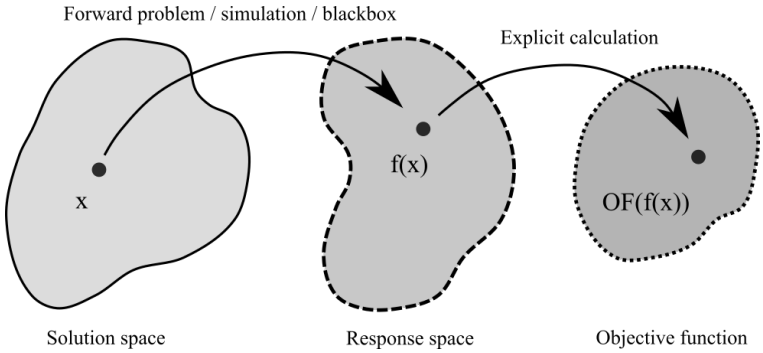
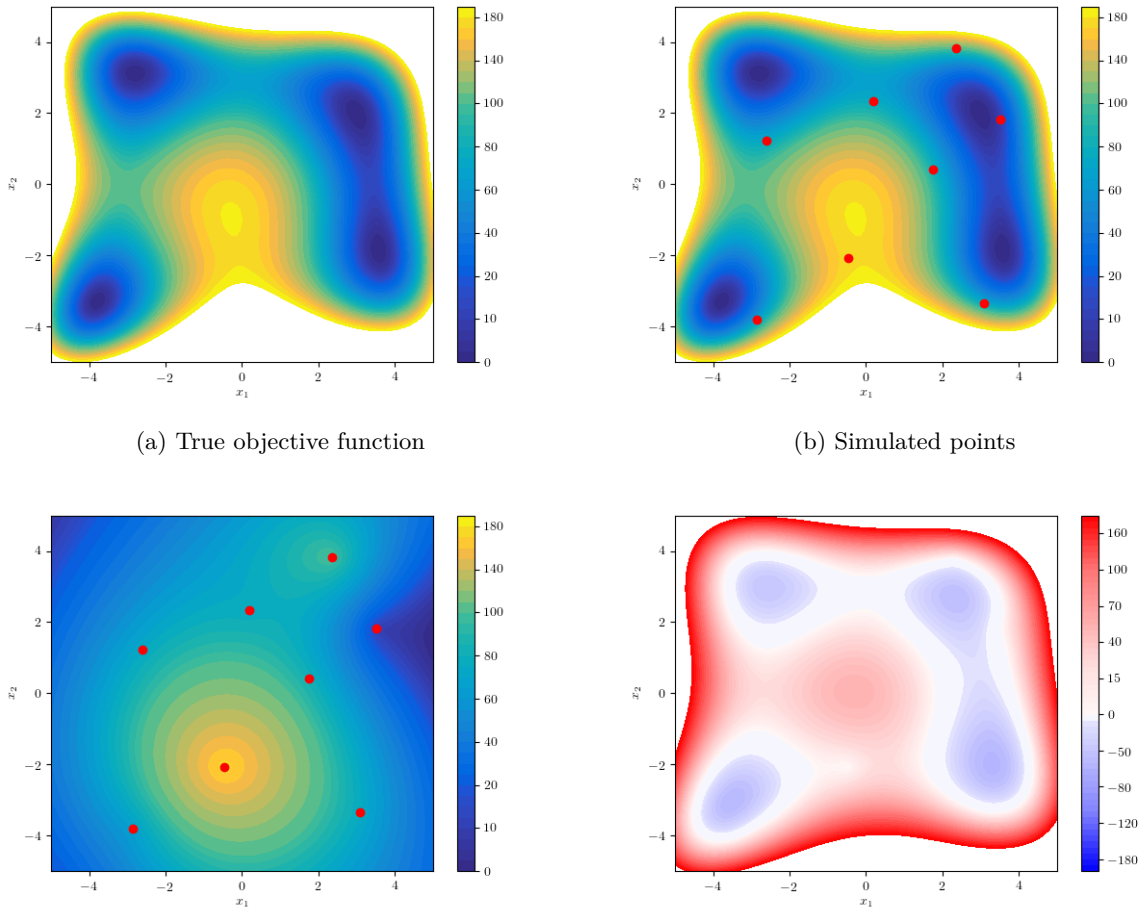


Figure 1: Illustration showing the forward problem. A set of parameters, x, is sampled from the solution space. The forward problem, $f(x)$, is calculated and from the results of the forward problem, an objective function is calculated using an explicit formula.

This could in theory be achieved through trial and error or by rigorously trying every possible combination of parameters until a satisfactory match is reached. For problems where simulating responses is not a trivial task, but computationally expensive neither option is feasible. The objective function is a *likelihood* function, which measures how likely it is a given set of parameters will result in a fit to the observed data. To overcome

the computational feasibility issue the logical choice is then to use an optimization algorithm to apply a systematic search of the solution space. Unfortunately most optimization algorithms still require a substantial amount of function calls in order to progress. The solution to this is creating an alternative forward model - a so-called *proxy model* - which is cheaper to evaluate and optimize on. There exists multiple different methods for generating proxy models. What they all have in common is that they provide a computationally inexpensive way to interpolate between a limited number of simulations of the actual forward problem.



(a) True objective function

(b) Simulated points

(c) Proxy-model

(d) Difference

Figure 2: (a): Underlying objective function. (b): Simulated points. (c): Proxy-model of objective function calculated based on simulated points. (d): Difference between the underlying objective function and the proxy-model.

Logically, the more actual solutions of the forward problem that are solved, the more robust the proxy model becomes. Conversely, the more time consuming it is to calibrate it. Typically the cost of running an optimization algorithm on the proxy model is insignificant relative to the cost of a simulation. The goal is then to apply a proxy model and an optimization approach which most efficiently searches the solution space of the underlying problem, while requiring a minimum of simulations to be run.

As mentioned the goal is not to converge towards a single deterministic optimum, but rather to characterize the uncertainty space. As such, finding a global optimum is not of paramount importance. Instead the goal is to efficiently characterize all areas of high likelihood, both local and global. This is achieved through an optimization method known as Bayesian Optimization. This method balances the dual purpose of refining areas of interest while also scoping out unexplored parts of the domain.

The model calibration process is split into three steps. Due to the *Curse of Dimensionality* it is important to limit the number of parameters to only the most impacting set. This subset of parameters is typically not known a priori. A *Screening* is launched to systematically investigate the impact of each parameter. Based on the results, the parameters are ranked by impact and a subset carried forward into the calibration process. Once the relevant set of parameters is defined, an *Investigation* is initiated in order to appropriately sample the entire solution space. This step is required to obtain a satisfactory initial calibration of the proxy model. The last step, *Refinement*, is an iterative process. An optimization algorithm is run on the current proxy model in order to identify areas of interest. An appropriate number of parameter combinations are sampled based on a so-called proposal function and a new set of simulations are launched with the sampled parameters. The new responses are used to update the proxy model, and a new optimization is run. There is no predefined termination criteria for this process. Essentially it is continued until the proxy model is sufficiently refined to adequately describe the underlying joint probability density function.

1.1. Parameters

Prior to delving into the various steps of the model calibration process, it is important to define what is meant by parameters. When building a model, both the choice of model and the parameters of the model will often initially be based on some hard data. For Oil & Gas this could be porosity and permeability measurements made on cores. For a solar farm it might be the average sunlight in any given month. The key issue here is that many such measurements comes with uncertainty, especially when extrapolating away from control points. Hence what is meant by a parameter is actually a marginal probability density function. The set of parameters and their dependence structure (correlation) makes up what is known as the *prior* joint probability density function. The point of the model calibration process is then to gain insight into this joint probability density function by solving an inverse problem. Ultimately the more likely part of the *prior* distribution is discovered, resulting in what is known as the *posterior* distribution which is the result of the process.

As for any optimization problem it is preferable if the function is convex and behaves monotonically with regards to the input parameters. This is typically difficult to achieve for complex physical systems which may observe oscillations or similar highly non-linear behavior. Further, it is even more difficult to achieve for objective functions where changes in parameters may have a positive impact on one simulation output, but an adverse effect on another. For continuous physical properties this criteria is often satisfied to the degree possible. For discrete properties where the order of input can be switched, it is important to verify that the input parameters effect on the output is at least approximately monotonic. The more monotonic the objective function behaves the fewer simulations are required to calibrate the proxy-model and the more robust it will be.

1.2. Screening

There is no defined approach to screen the initial set of parameters. How rigorous it has to be depends largely on how well understood the problem is prior to embarking on it. It also depends on the degree of non-linearity of the problem.

For the purpose of comparing the impact of various parameters it is convenient to use a proxy-model from the polynomial family (linear, quadratic, etc.). For simple problems it is sufficient to rank the linear terms to quantify the impact, in which case a classical sensitivity analysis (One Variable at a Time) is fit for purpose. In non-linear problems it might be required to include an arbitrary number of interaction terms as well, requiring a factorial design, and in highly non-linear problems higher order terms may even be necessary, requiring a design such as a central composite design in order to measure the curvature.

There can be an order of magnitude difference in the amount of required simulations between the de-

sign so naturally careful considerations has to be given up front. On the other hand if the chosen approach is too simplistic it may subsequently prove impossible to obtain a satisfactory model calibration because important parameters were excluded.

If the problem is completely understood upfront and all necessary parameters are known a priori, this step can be foregone.

1.3. Investigation

Once a representative subset of parameters have been selected it is necessary to investigate the interior of the solution space, not just the extrema. This is achieved by the use of a space-filling design which ensures parameters are sampled with sufficient distance to maximize the resulting information. Again, the number of required simulations depend on the dimension and the non-linearity of the problem. As described previously the optimization algorithm which searches for areas of interest during refinement is run on the proxy-model. If the proxy-model is not well calibrated initially, there is a risk of diverging towards areas of interest in the proxy-model which are not areas of interest of the underlying problem. This can result in a waste of time or even an incorrect solution to the problem.

1.4. Refinement

Refinement is the core of the matching process. During this iterative procedure the key areas of interest on the proxy-model are investigated and refined by launching new simulations. At each iteration the calibration of the proxy-model is improved making it more representative of the underlying problem. The reason for limiting the refinement to areas of high likelihood is in order to reduce the number of required simulations. The heuristic for proposing new simulations is a key aspect of `modpy` as this is what provides the maximum amount of information in the minimum number of simulations.

2. Optimization

One of the primary reasons for constructing a model of a physical system is to optimize a set of design parameters. This process is similar to the Model Calibration phase, in the sense that a set of parameters are optimized by solving an inverse problem. The difference is that the objective function no longer measures the discrepancy between simulations and observed data, but instead some engineering or financial value which is maximized. There is no theoretical limitation on the number of design parameter, but there is a practical one both in terms of computational limitations and engineering practicalities. The method still suffers from the *Curse of Dimensioniality* which puts a limit on the number of parameters that can be investigated in a reasonable time-frame. Also, most problems can be limited by practical engineering constraints related to manufacturing or similar.

2.1. Solution Space Reduction

The parameter space can often be limited by external factors, or even be rephrased in a simple formulation by imposing logical assumptions. Occasionally, the limited solution space is still too vast for the problem to be solved in reasonable time. In such situations it is necessary to apply a different type of optimizer which can reduce the solution space based on qualified assumptions. These types of pre-solvers are beyond the scope of this module as they will need to be problem specific to be effective. An example of such a pre-solver could be optimizing the position of wells in an oil reservoir using a time-of-flight calculation of the geological model, as opposed to a full reservoir simulation. There is a risk that such a pre-solver will limit the solution space to not include the true global optimum. This is why the pre-solver has to be problem specific and used with care.

2.2. Fixed & Variable Parameters

In engineering applications design variables can often be split into two categories. Parameters that are defined prior to development and fixed throughout the lifetime of a project, product, etc., and parameters which can be changed at various stages of the lifetime. Examples of the former are wells in a reservoir or the design of an airfoil. The latter type of parameters are things such as valve settings. Fixed parameters can be optimized using conventional problem formulations, but variable parameters are most suited to be solved with an approach called Model Predictive Control.

2.3. Optimization under Uncertainty

When the model is uncertain it is not guaranteed that the optimal set of design parameters is the same for all possible realizations of the model. To deal with this ambiguity an option is to find a set of design parameters which maximizes some probabilistic measurement across the possible model realizations. This approach can in many ways be thought of as managing a portfolio of stocks. Each design parameter corresponds to a stock and the objective function for each realization is the return. Dependent on the objective the "portfolio" can be optimized to maximize the expected return, minimize the potential downside or similar. This is done using an ensemble based approach, where an ensemble of realizations is sampled from the posterior distribution (or prior if no calibration is done). This ensemble is then assumed representative of the entire distribution and used to calculate statistical properties such as means, percentiles, etc. of the objective function.

3. Forecasting

For engineering application such as airfoil or engine optimization the process stops at the optimization stage. However, for development projects, such as energy projects (oil & gas, wind, solar, ...), the associated energy production has to be forecasted. Using the calibrated posterior distribution and the optimal design parameters an ensemble of production results can be simulated. If the design parameters were optimized under uncertainty the same ensemble is used, alternatively an ensemble is sampled. Each realization in the ensemble is simulated and statistical values are extracted. Based on the statistical results investment decisions can be made balancing the risk and reward.

Part II
Theory

4. Random Sampling

4.1. Inverse Transform Sampling

Random sampling from defined distributions is done via a method called inverse transform sampling. This method requires knowledge of the cumulative distribution function of the sampled distribution.
Let $F(x)$, $x \in \mathbb{R}$ denotes any cumulative distribution function. Consequently, the mapping $F : \mathbb{R} \to [0, 1]$ is a non-negative and monotone (non-decreasing) function that is continuous from the right (i.e. for non-negative numbers) and has the left hand limits $F(x) \in [0, 1]$; furthermore, $F(\infty) = 1$ and $F(-\infty) = 0$. The goal is to generate a random value sample X with the cumulative distribution function F, i.e. to simulate X such that $\Pr[X \leq x] = F(x)$, $x \in \mathbb{R}$.
Define the generalized inverse of F, $F^{-1} : [0, 1] \to \mathbb{R}$ as

$$F^{-1}(y) = \min\{x : F(x) \geq y\}, \, y \in [0, 1] \tag{1}$$

Given that F is continuous over the domain it is also invertible, in which case a stricter definition holds for (1), namely $F^{-1}(y) = \min\{x : F(x) = y\}$. Thus $F(F^{-1}(y)) = y$ and $F^{-1}(F(x)) = x$ with $F^{-1}(y)$ similar to F being a monotone function in y. All this is to say that given a known cumulative distribution function, F, there exists an inverse mapping which can transform a uniformly distributed number, $y \in [0, 1]$ into a sample, x, which follows the distribution of F, as described by the following lemma.

Lemma 4.1 (Inverse Transform Sampling)**.**
Let $F(x)$, $x \in \mathbb{R}$, denote any cumulative distribution function. Let $F^{-1}(y)$, $y \in [0, 1]$ denote the inverse function defined in (1)*. Define $X = F^{-1}(U)$, where U has the continuous uniform distribution over the interval* $(0, 1)$*. Then X is dstributed as F, that is, $Pr[X \leq x] = F(x)$, $x \in \mathbb{R}$.*

The proof can be found in [1].

4.2. Multivariate Sampling

Often when sampling random values for stochastic modelling samples have to be drawn from multiple distribution functions simultaneously. These individual distributions will typically have a shared dependency structure. The most typical measure of dependence is Pearson's Correlation Coefficient, which is defined in terms of the product of the statistical moments of two random samples, X and Y

$$\rho_{X,Y} = \text{Corr}[X,Y] = \frac{\text{Cov}[X, Y]}{\sigma_X \sigma_Y} \tag{2}$$

where Corr is the correlation, Cov is covariance and σ is the standard deviation. $\rho \in [-1, 1]$ describes the linear dependence between X and Y. This linearity causes an issue as non-linear transforms of X and Y does not preserving the linear correlation structure. That is not to say they are no longer correlated after the transform, but simply that $\text{Corr}[X, Y] \neq \text{Corr}\left[F^{-1}(X), F^{-1}(Y)\right]$.

Example 4.2 (Correlated Lognormal Distributions)**.**
Let $(Z_1, Z_2) \sim \mathcal{N}\left(0, \Sigma^2\right)$ be a multivariate normal distribution with variance $\sigma^2_{i=j} = 1$ and $\sigma^2_{i \neq j} = \rho$. Assuming Z_1 represents the underlying normal distribution of a lognormal distribution then $X_1 = e^{Z_1}$ and similarly $X_2 = e^{Z_2}$ are lognormally distributed. The exponential function is non-linear and consequently does not preserve the linear correlation. It can be shown that the Pearson's correlation for X_1 and X_2 after the

transform is given by

$$\text{Corr}[X_1, X_2] = \frac{e^\rho - 1}{e - 1} \tag{3}$$

with the following consequence

$$\begin{cases} \text{Corr}[X_1, X_2] < \text{Corr}[Z_1, Z_2], & \text{for } \rho < 1 \\ \text{Corr}[X_1, X_2] = \text{Corr}[Z_1, Z_2], & \text{for } \rho = 1 \end{cases}$$

i.e. strictly less than the underlying normal distributions correlations for $\rho < 1$. Given this specific case where (3) is known, it is possible to solve for ρ to determine the required Pearson's correlation to supply to the underlying normal distribution in order to have the correct linear correlation between X_1 and X_2, namely

$$\rho = \ln\left(\text{Corr}[X_1, X_2](e - 1) + 1\right) \tag{4}$$

however, an exact formula for the for the corresponding correlation as seen in (4) is generally very difficult to determine and sometimes impossible. Further, it can be shown for the case of (4) that the solution is not bounded in $\rho \in [-1, 1]$ as expected, but rather $\rho \in [-1/e, 1]$ thus not all linear correlations are attainable [2].

Given that many cumulative inverse transforms are non-linear, (2) is not an appropriate measure of dependency for general multivariate sampling. In order to avoid being limited to dependency preserving transforms, it is desirable to find a measure of correlation which is transform independent and this is where rank correlations, such as Spearman's, ρ_s and Kendall's, τ prove useful.

This leads to the phenomenon of copulas. A copula essentially describes a multivariate distribution on the unit hypercube with uniform marginal distributions, that is a multivariate distribution of which all the marginal distributions are uniform, which is exactly what is required as input to the cumulative inverse transform.

Let $(X_1, X_2, ..., X_n)$ be a multivariate distribution, with a similar sequence of marginal distribution functions, $F_i(x) = \Pr[X_i \leq x]$, which are continuous. By applying the Probability Integral Transform (the opposite transform of the Inverse Cumulative Transform) a new multivariate distribution is obtained, $(U_1, U_2, ..., U_n) = (F_1(X_1), F_2(X_2), ..., F_n(X_n))$ of which all the marginal distribution functions are uniform distributions on the interval $u \in [0, 1]$. It is said that the copula of the sequence $(X_1, X_2, ..., X_n)$ is the joint cumulative distribution function of $(U_1, U_2, ..., U_n)$

$$C(u_1, u_2, ..., u_n) = \Pr[U_1 \leq u_1, U_2 \leq u_2, ..., U_n \leq u_n] \tag{5}$$

the copula, C, holds all the information about the dependence structure between the individual components of $(X_1, X_2, ..., X_n)$ whereas the marginal cumulative distributions functions, F_i, holds all the information about the marginal distributions. Further, by the definition of Lemma 4.1 the inverse transform allows for rewriting (5) to

$$C(u_1, u_2, ..., u_n) = \Pr\left[X_1 \leq F_1^{-1}(u_1), X_2 \leq F_2^{-1}(u_2), ..., X_n \leq F_n^{-1}(u_n)\right]$$

There exists many different well defined copulas which can broadly be split into three categories

- Archimedean copulas
- Elliptical copulas
- Vine copulas

In multivariate space Archimedean copulas only allow for a single measure of dependency among the sequence, i.e. $\rho_{i,j} = \rho$, which is undesirable when describing dependency between properties in a physical system. Elliptical copulas such as the Gaussian copula and t-copula (from Student's T-distribution) allows for defining

individual correlations between the various components of the sequence, but the dependence cannot be asymmetrical, i.e. $\rho_{i,j} = \rho_{j,i}$. The point about asymmetry is what vine copulas are able to deal with, namely they can handle dependence structures where $\rho_{i,j} \neq \rho_{j,i}$. The requirement of asymmetrical dependence structures is something that arises in financial portfolio management, but rarely a requirement when modelling physical systems. As such the focus will be on elliptical copulas and more specifically the Gaussian copula.

4.2.1. Sampling from a Copula

Define the Gaussian copula, $C_{\Sigma}^{\text{Gauss}}$, as the joint cumulative distribution function of the Gaussian distribution, $\Phi_{\Sigma} \sim \mathcal{N}(0,\Sigma)$ where Σ is the linear correlation structure between each component in the joint normally distributed distribution function, then

$$C_{\Sigma}^{\text{Gauss}}(u_1, u_2, .., u_n) = \Phi_{\Sigma}\left(\Phi^{-1}(u_1), \Phi^{-1}(u_2), ..., \Phi^{-1}(u_n)\right) \tag{6}$$

from (6) it is evident that the Gaussian copula is given by the joint cumulative distribution function of the multivariate normal distribution, Φ_{Σ} and each of the marginal distribution functions are given by the cumulative standard normal distribution function, Φ. At this point it is possible to combine the learnings from Section 4.1 and Section 4.2. By sampling from the multivariate normal distribution it is possible to obtain a sequence, $(Z_1, Z_2, ..., Z_n) \sim \mathcal{N}(0,\Sigma)$. Given that the cumulative normal distribution is continuous and monotone applying Lemma 4.1 allows for transforming the marginal distribution of each of the components in the sequence to a uniform distribution, $(U_1, U_2, ..., U_n) = (\Phi^{-1}(Z_1), \Phi^{-1}(Z_2), ..., \Phi^{-1}(Z_n))$ while preserving the dependence structure of the joint distribution in the copula. Given a sequence of uniformly distributed random variables, each of the can be inversely transformed to an arbitrary distribution with a well defined, continuous and monotone cumulative distribution function, i.e. $(X_1, X_2, ..., X_n) = (F_1^{-1}(U_1), F_2^{-1}(U_2), ..., F_n^{-1}(U_n))$, with each component distributed according to the their respective cumulative distribution functions, F_i and the dependence structure still preserved in the copula.

As mentioned the dependence structure input to the copula, Σ, is the linear correlation, i.e. Pearson's correlation coefficient, which is mentioned is not preserved through non-linear transformation. Assuming the user-supplied correlation matrix for the components in the sequence $(X_1, X_2, ..., X_n)$ is in fact the rank correlation coefficients then a transform to the linear correlation structure is required prior to sampling. For the multivariate normal there exists a set of good approximations for mapping, namely

$$\tau = \frac{2}{\pi}\arcsin(\rho), \quad \rho = \sin\left(\tau\frac{\pi}{2}\right)$$

and

$$\rho_s = \frac{6}{\pi}\arcsin\left(\frac{\rho}{2}\right), \quad \rho = 2\sin\left(\rho_s\frac{\pi}{6}\right)$$

By use of the method outlined above it is possible to sample correlated random variables with arbitrary marginal distribution functions while approximately preserving the dependence structure.

4.3. Pseudo Random Number Generation

In order to use the inverse transform sampling method, a method is required for sampling uniform distributed values on the interval $u \in (0, 1)$. For this a pseudo random number generator (PRNG) is used, which based on an initial seed value and a defined period generates a sample of seemingly random numbers. The most common algorithm used is the Mersenne Twister algorithm. This is the default PRNG used in software such as Excel, Matlab, Python, R and many more. Mersenne Twister samples uniformly distributed random numbers over the interval, $u \in [0, 1)$.

4.4. Distributions

This sections presents a series of distributions as well as the logarithmic and root transforms of these distributions. For the normal distribution, the logarithmic transform of the distribution is shown both for the well known lognormal distribution (using the natural logarithm) and the more generic case. For the distributions with finite support it can be shown that the logarithmic transforms are independent of the base. For these distributions only the generic case will be shown.

4.4.1. Normal Distribution

Parameters

The normal distribution takes two parameters - the first and second moment - mean, μ, and standard deviation, σ, where the standard deviation is the square-root of the variance. In engineering applications is it often more convenient to derive theoretical limits (minimum and maximum). In such a case the supplied extremums may be used to calculate the first and second moment. Given that the PDF of the normal distribution has support on the entire range of real numbers, i.e. $f(x) \in \mathbb{R}$, a minimum and maximum is not theoretically supported for the normal distribution. This means an approximation is required in the calculation of μ and σ. For the normal distribution $\mu \pm \sigma$ accounts for ≈ 68 % of the uncertainty, $\mu \pm 2\sigma$ accounts for ≈ 95.4 % and $\mu \pm 3\sigma$ accounts for ≈ 99.7 %. Given that PRNGs typically are poor at sampling the tails of distributions, 99.7 % should capture a sufficient amount of the uncertainty. By that assumption, the mean and standard deviation can be defined explicitly as functions of the minimum and maximum

$$\mu = \frac{b+a}{2}, \quad \sigma = \frac{b-a}{6}$$

Notice that a common choice is to use $\mu \pm 2\sigma$ and consequently calculate σ by division by 4.

Linear Transform

The PDF of the normal distribution is given by

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad x \in \mathbb{R}$$

with parameters $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}_+$. The CDF is

$$F(x;\mu,\sigma) = \frac{1}{2}\left(1 + \operatorname{Erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right), \quad x \in \mathbb{R}$$

and the inverse CDF is

$$x = F^{-1}(y;\mu,\sigma) = \mu + \sigma\sqrt{2} \cdot \operatorname{Erf}^{-1}(2y-1), \quad y \in [0,1]$$

For convenience the standard normal distribution is introduced as well, which is defined as the normal distribution with $\mu = 0$ and $\sigma = 1$. The PDF is

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} \Leftrightarrow f(x;\mu,\sigma) = \frac{1}{\sigma}\phi\left(\frac{x-\mu}{\sigma}\right)$$

the CDF

$$\Phi(z) = \frac{1}{2}\left(1 + \operatorname{Erf}\left(\frac{z}{\sqrt{2}}\right)\right) \Leftrightarrow F(x;\mu,\sigma) = \Phi\left(\frac{x-\mu}{\sigma}\right)$$

and the inverse CDF

$$x = \Phi^{-1}(y) = \sqrt{2} \cdot \operatorname{Erf}^{-1}(2y-1) \Leftrightarrow x = F^{-1}(y;\mu,\sigma) = \mu + \sigma \cdot \Phi^{-1}(y)$$

where z is a normalized variable. The formulation of the standard normal distribution is convenient as all transforms of the normal distribution can be written as a function of the standard normal distribution with normalized input.

Log Transform (Natural)

The logarithmic transform of the normal distribution is what is known as the lognormal distribution. Let Z be a standard normal distribution, then X is lognormally distributed if

$$X = e^{\mu+\sigma Z}, \quad Z \sim \mathcal{N}(0,1) \tag{7}$$

Here the parameters μ and σ does not correspond to the first and second moment of the lognormal distribution, but are derived from the first and second moment of the underlying normal distribution. Given the mean, μ_X, and variance , σ_X^2, of X the corresponding μ and σ of the underlying normal distribution may be calculated by

$$\begin{aligned} \mu &= \ln\left(\frac{\mu_X^2}{\sqrt{\mu_X^2+\sigma_X^2}}\right) \\ \sigma &= \sqrt{\ln\left(1+\frac{\sigma_X^2}{\mu_X^2}\right)} \end{aligned} \tag{8}$$

for $\mu_X \in \mathbb{R}_+$ and $\sigma_X \in \mathbb{R}_+$. The PDF is

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\ln(x)-\mu}{\sigma}\right)^2} = \frac{1}{x\sigma}\phi\left(\frac{\ln(x)-\mu}{\sigma}\right), \quad x \in \mathbb{R}_+$$

The CDF is

$$F(x;\mu,\sigma) = \frac{1}{2}\left(1+\operatorname{Erf}\left(\frac{\ln(x)-\mu}{\sigma\sqrt{2}}\right)\right) = \Phi\left(\frac{\ln(x)-\mu}{\sigma}\right), \quad x \in \mathbb{R}_+$$

and the inverse CDF is

$$x = F^{-1}(y;\mu,\sigma) = e^{\mu+\sigma\sqrt{2}\cdot\operatorname{Erf}^{-1}(2y-1)} = e^{\mu+\sigma\Phi^{-1}(y)}, \quad y \in [0,1]$$

Log Transform (Generic)

The estimate of (8) for the underlying distribution of (7) holds true not only for $\ln(X)$, but any logarithmic base, i.e. if $\log_n(X)$ is normally distributed, then so is $\log_m(X)$ for any numbers, n, $m \neq 1$. Similarly, if e^Y is log-normally distributed, then so is n^Y. This means that the distribution for any kind of log transform, can be estimated in a similar fashion, namely

$$X = n^{\mu+\sigma Z}, \quad Z \sim \mathcal{N}(0,1)$$

Given a desired mean, μ_X, and variance , σ_X^2, of X the required μ and σ of the underlying normal distribution may be calculated by

$$\begin{aligned} \mu &= \log_n\left(\frac{\mu_X^2}{\sqrt{\mu_X^2+\sigma_X^2}}\right) \\ \sigma &= \sqrt{\log_n\left(1+\frac{\sigma_X^2}{\mu_X^2}\right)} \end{aligned} \tag{9}$$

for $\mu_X \in \mathbb{R}_+$ and $\sigma_X \in \mathbb{R}_+$. The PDF is

$$f(x;\mu,\sigma) = \frac{1}{x\ln(n)\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log_n(x)-\mu}{\sigma}\right)^2} = \frac{1}{x\sigma\ln(n)}\phi\left(\frac{\log_n(x)-\mu}{\sigma}\right), \quad x \in \mathbb{R}_+$$

the CDF is

$$F(x;\mu,\sigma) = \frac{1}{2}\left(1+\operatorname{Erf}\left(\frac{\log_n(x)-\mu}{\sigma\sqrt{2}}\right)\right) = \Phi\left(\frac{\log_n(x)-\mu}{\sigma}\right), \quad x \in \mathbb{R}_+$$

and the inverse CDF is

$$x = F^{-1}(y;\mu,\sigma) = n^{\mu+\sigma\sqrt{2}\cdot\mathrm{Erf}^{-1}(2y-1)} = n^{\mu+\sigma\Phi^{-1}(y)}, \quad y \in [0,1]$$

The natural case and generic case are the same when the base, $n = e$[1].

Root Transform

A variable, X, is said to be root-normal distributed if

$$X = (\mu + \sigma Z)^n, \quad Z \sim \mathcal{N}(0,1)$$

Similarly to the log-transform, the parameters μ and σ does not correspond to the moments of the transformed distribution, but rather the underlying normal distribution, Z. The calculation of the underlying parameters is not trivial for any $n \in \mathbb{N}_+$. It is provided for $n = 2$, i.e. the square-root transform. μ and σ is calculated in a similar fashion as (8), see Appendix 14.5

$$\begin{aligned} \mu_{n=2} &= \sqrt{\mu_X^2 - \frac{\sigma_X^2}{2}} \\ \sigma_{n=2} &= \sqrt{\mu_X - \sqrt{\mu_X^2 - \frac{\sigma_X^2}{2}}} \end{aligned} \tag{10}$$

for $\mu_X \geq \sqrt{\frac{\sigma_X^2}{2}}$ and $\sigma_X \in \mathbb{R}_+$. The PDF is

$$f(x;\mu,\sigma) = \frac{x^{\frac{1}{n}-1}}{\sigma n\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\sqrt[n]{x}-\mu}{\sigma}\right)^2} = \frac{x^{\frac{1}{n}-1}}{n\sigma}\phi\left(\frac{\sqrt[n]{x}-\mu}{\sigma}\right), \quad x \in \mathbb{R}_{\geq 0}$$

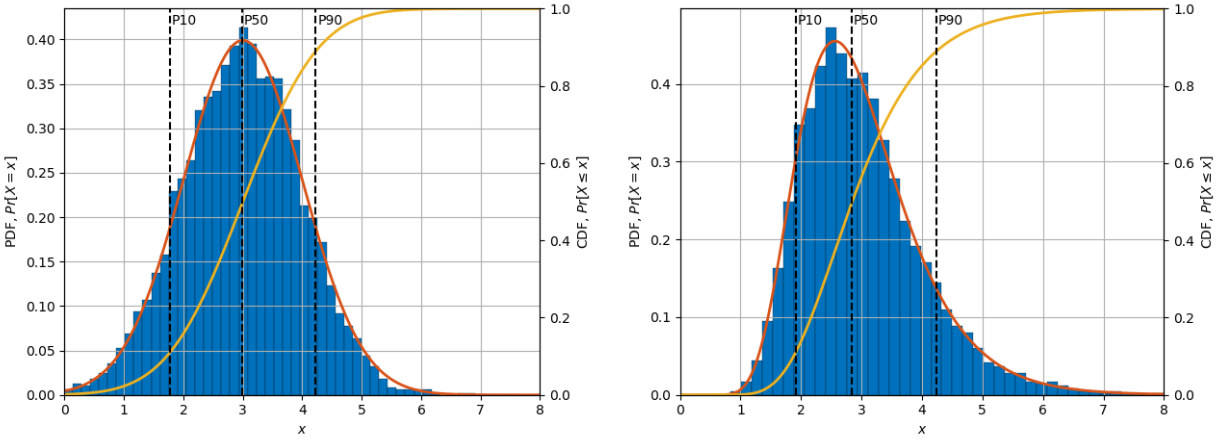
The CDF is

$$F(x;\mu,\sigma) = \frac{1}{2}\left(1 + \mathrm{Erf}\left(\frac{\sqrt[n]{x}-\mu}{\sigma\sqrt{2}}\right)\right) = \Phi\left(\frac{\sqrt[n]{x}-\mu}{\sigma}\right), \quad x \in \mathbb{R}_{\geq 0}$$

and the inverse CDF is

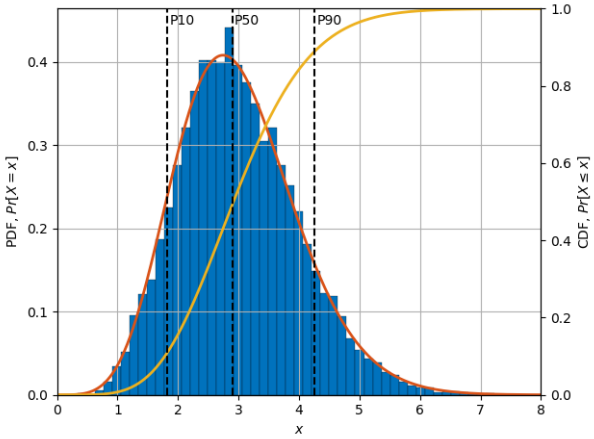
$$x = F^{-1}(y;\mu,\sigma) = \left(\mu + \sigma\sqrt{2}\cdot\mathrm{Erf}^{-1}(2y-1)\right)^n = \left(\mu + \sigma\cdot\Phi^{-1}(y)\right)^n, \quad y \in [0,1]$$

Examples



(a) Linear

(b) Log



(c) Square-root

Figure 3: Examples of variations of the normal distribution with $\mu_X = 3$ and $\sigma_X = 1$.

4.4.2. Truncated Normal Distribution

Parameters

The truncated normal distribution takes the same input parameters as the normal distribution. As mentioned previously the normal distribution has support in $x \in [-\infty, \infty]$, consequently any PRNG can in practice sample outside the range of the defined minimum and maximum. To overcome this issue the truncated normal distribution ensures support in the interval $x \in [a, b]$.

Linear Transform

The PDF of the truncated normal distribution is

$$f(x; \mu, \sigma, a, b) = \frac{1}{\sigma} \frac{\phi\left(\frac{x-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)}$$

where $\phi(\cdot)$ is the PDF of the standard normal distribution and $\Phi(\cdot)$ is the CDF of the standard normal distribution. The CDF is

$$F(x; \mu, \sigma, a, b) = \frac{\Phi\left(\frac{x-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)}$$

and the inverse CDF is

$$x = F^{-1}(y; \mu, \sigma, a, b) = \Phi^{-1}\left(\Phi\left(\frac{a-\mu}{\sigma}\right) + y \cdot \left(\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)\right)\right)\sigma + \mu$$

where $\Phi^{-1}(\cdot)$ is the inverse CDF of the standard normal distribution. Notice that the inverse CDF of the truncated normal distribution is not used for sampling as it can cause numerical difficulties in the tails. Instead a Rejection-Sampling method is used [6].

Log Transform (Generic)
A variable, X, is said to follow a truncated log-normal distribution if

$$X = n^Z, \quad Z \sim \mathcal{N}(\mu, \sigma, \log_n(a), \log_n(b)) \tag{11}$$

where μ and σ are given by (9). The PDF of the truncated lognormal distribution is

$$f(x; \mu, \sigma, a, b) = \frac{1}{x \ln(n) \sigma} \cdot \frac{\phi\left(\frac{\log_n(x)-\mu}{\sigma}\right)}{\Phi\left(\frac{\log_n(b)-\mu}{\sigma}\right) - \Phi\left(\frac{\log_n(a)-\mu}{\sigma}\right)}$$

The CDF is

$$F(x; \mu, \sigma, a, b) = \frac{\Phi\left(\frac{\log_n(x)-\mu}{\sigma}\right) - \Phi\left(\frac{\log_n(a)-\mu}{\sigma}\right)}{\Phi\left(\frac{\log_n(b)-\mu}{\sigma}\right) - \Phi\left(\frac{\log_n(a)-\mu}{\sigma}\right)}$$

and the inverse CDF is

$$x = F^{-1}(y; \mu, \sigma, a, b) = \Phi^{-1}\left(\Phi\left(\frac{\log_n(a)-\mu}{\sigma}\right) + y \cdot \left(\Phi\left(\frac{\log_n(b)-\mu}{\sigma}\right) - \Phi\left(\frac{\log_n(a)-\mu}{\sigma}\right)\right)\right)\sigma + \mu$$

Sampling from the truncated lognormal distribution is done by using the expression (11) where Z is sampled using the same approach as outlined in the linear transform.

Root Transform
A variable, X, is said to follow a truncated root-normal distribution if

$$X = Z^n, \quad Z \sim \mathcal{N}(\mu, \sigma, \sqrt[n]{a}, \sqrt[n]{b}) \tag{12}$$

where μ and σ are given by (10) which are only viable for $n = 2$. The PDF of the truncated root-normal distribution is

$$f(x; \mu, \sigma, a, b) = \frac{x^{\frac{1}{n}-1}}{n\sigma} \cdot \frac{\phi\left(\frac{\sqrt[n]{x}-\mu}{\sigma}\right)}{\Phi\left(\frac{\sqrt[n]{b}-\mu}{\sigma}\right) - \Phi\left(\frac{\sqrt[n]{a}-\mu}{\sigma}\right)}$$

The CDF is

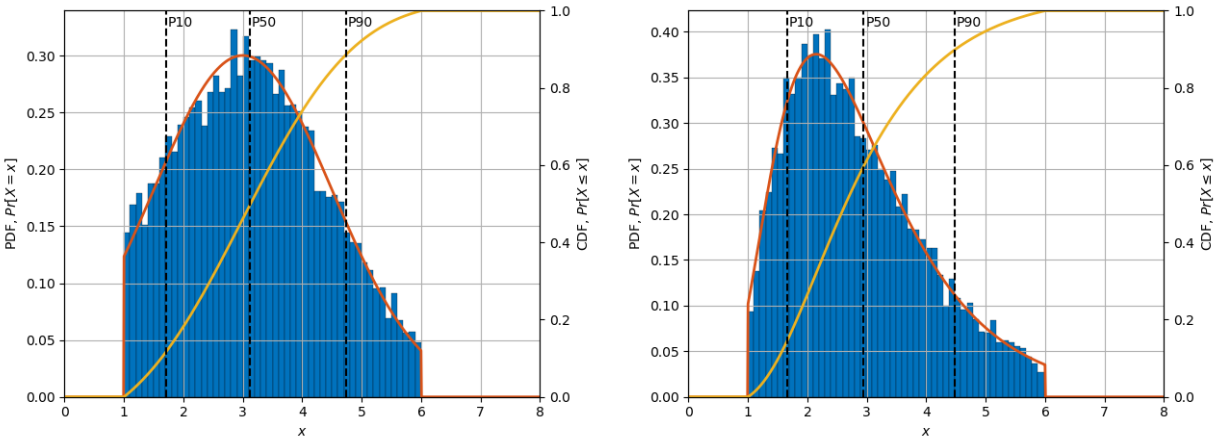
$$F(x; \mu, \sigma, a, b) = \frac{\Phi\left(\frac{\sqrt[n]{x}-\mu}{\sigma}\right) - \Phi\left(\frac{\sqrt[n]{a}-\mu}{\sigma}\right)}{\Phi\left(\frac{\sqrt[n]{b}-\mu}{\sigma}\right) - \Phi\left(\frac{\sqrt[n]{a}-\mu}{\sigma}\right)}$$

and the inverse CDF is

$$x = F^{-1}(y; \mu, \sigma, a, b) = \Phi^{-1}\left(\Phi\left(\frac{\sqrt[n]{a}-\mu}{\sigma}\right) + y \cdot \left(\Phi\left(\frac{\sqrt[n]{b}-\mu}{\sigma}\right) - \Phi\left(\frac{\sqrt[n]{a}-\mu}{\sigma}\right)\right)\right)\sigma + \mu$$

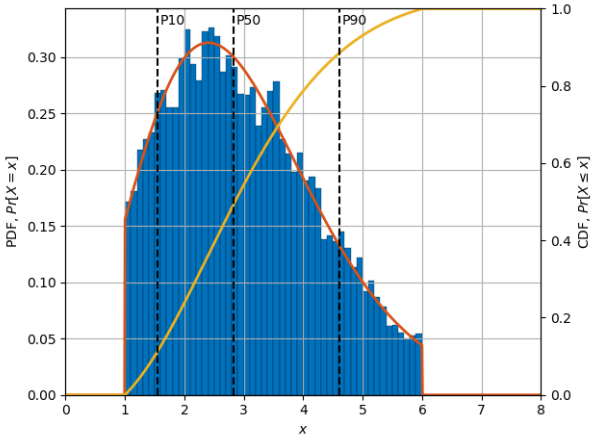
Sampling from the truncated root-normal distribution is done by using the expression (12) where Z is sampled using the same approach as outlined in the linear transform.

Examples



(a) Linear

(b) Log



(c) Square-root

Figure 4: Examples of variations of the normal distribution with $\mu_X = 3$ and $\sigma_X = 1.5$, $a = 1$ and $b = 6$.

4.4.3. Uniform Distribution

Parameters

The uniform distribution takes the minimum and maximum as input, so no calculation of parameters is required.

Linear Transform

The PDF of the uniform distribution is given by

$$f(x; a, b) = \begin{cases} \frac{1}{b-a}, & \text{for } x \in [a,\, b] \\ 0 & \text{otherwise} \end{cases}$$

and the CDF

$$F(x; a, b) = \begin{cases} 0, & \text{for } x < a \\ \frac{x-a}{b-a}, & \text{for } x \in [a,\, b] \\ 1, & \text{for } x > b \end{cases}$$

the inverse cumulative distribution function for mapping between a uniform distribution supported in $y \in [0,\,1]$ and a uniform distribution supported in $y \in [a,\,b]$ is

$$x = a + (b-a)y, \quad y \in [0,1]$$

Log Transform

The log-uniform distribution is also called the reciprocal distribution. A variable, X, is said to follow a log-uniform distribution if

$$X = e^Z, \quad Z \sim U\left(\ln(a), \ln(b)\right)$$

Similar to the lognormal distribution the statement is true for any logarithmic base, n. However, it can be shown that the log-uniform distribution is unaffected by the choice of logarithmic base, n, see Appendix 14.6 for proof. Therefore it is assumed that the logarithmic base corresponds to the natural logarithm, as this simplifies the expressions. Unlike the normal distribution, the parameters used to define the uniform distribution are invariant to a monotone transform such as the logarithmic and root transform, i.e.

$$x_j = \min\left(\{x_i\}_{i=0}^n\right) \Leftrightarrow g(x_j) = \min\left(\{g\left(x_i\right)\}_{i=0}^n\right)$$

where $g(\cdot)$ is a monotone transform. This similarly holds true for the maximum. Therefore the PDF can be expressed in terms of the PDF of the uniform distribution, in this case denoted f_Z, the PDF is

$$f_X(x;a,b) = f_Z\left(\ln(x);\ln(a),\ln(b)\right)\cdot\frac{1}{x} = \begin{cases} \frac{1}{x\ln\left(\frac{b}{a}\right)}, & \text{for } x \in [a,\,b] \\ 0 & \text{otherwise} \end{cases}$$

The CDF is

$$F_X(x;a,b) = F_Y\left(\ln(x);\ln(a),\ln(b)\right) = \begin{cases} 0, & \text{for } x < a \\ \log_{\frac{b}{a}}\left(\frac{x}{a}\right), & \text{for } x \in [a,\,b] \\ 1, & \text{for } x > b \end{cases}$$

and the inverse CDF is

$$x = F_X^{-1}(y;a,b) = e^{F_Y^{-1}(y;\ln(a),\ln(b))} = e^{\ln\left(\frac{b}{a}\right)y+\ln(a)}, \quad y \in [0,1]$$

Root Transform

A variable, X, is said to follow a root-uniform distribution if

$$X = Z^n, \quad Z \sim U\left(\sqrt[n]{a},\,\sqrt[n]{b}\right)$$

where n is the root. The PDF of the root-uniform distribution is

$$f(x;a,b) = f_Y\left(\sqrt[n]{x};\,\sqrt[n]{a},\,\sqrt[n]{b}\right)\cdot\frac{1}{n}x^{\frac{1}{n}-1} = \begin{cases} \frac{1}{n(\sqrt[n]{b}-\sqrt[n]{a})}x^{\frac{1}{n}-1}, & \text{for } x \in [a,\,b] \\ 0 & \text{otherwise} \end{cases}$$

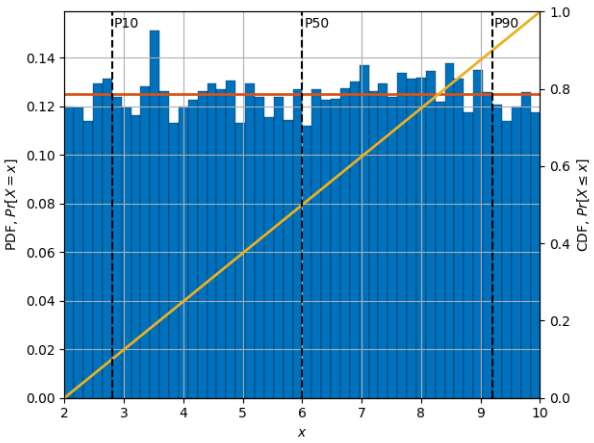
where n is the root. The CDF is simply

$$F(x;a,b) = F_Y\left(\sqrt[n]{x};\,\sqrt[n]{a},\,\sqrt[n]{b}\right) = \begin{cases} 0, & \text{for } x < a \\ \frac{\sqrt[n]{x}-\sqrt[n]{a}}{\sqrt[n]{b}-\sqrt[n]{a}}, & \text{for } x \in [a,\,b] \\ 1, & \text{for } x > b \end{cases}$$

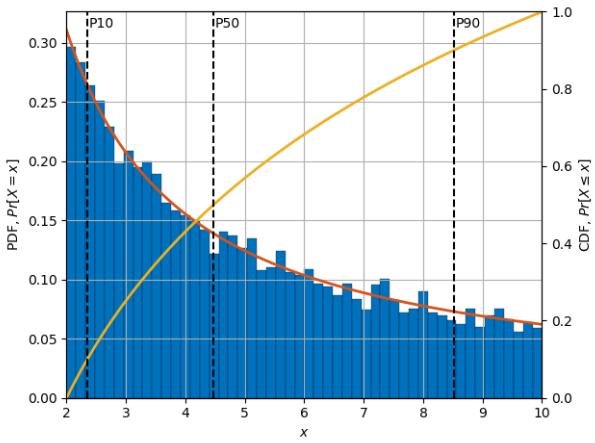
and the inverse CDF

$$x = F_X^{-1}(x;a,b) = \left(F_Y^{-1}\left(\sqrt[n]{x};\,\sqrt[n]{a},\,\sqrt[n]{b}\right)\right)^n = \left(y\left(\sqrt[n]{b}-\sqrt[n]{a}\right)+\sqrt[n]{a}\right)^n, \quad y \in [0,1]$$

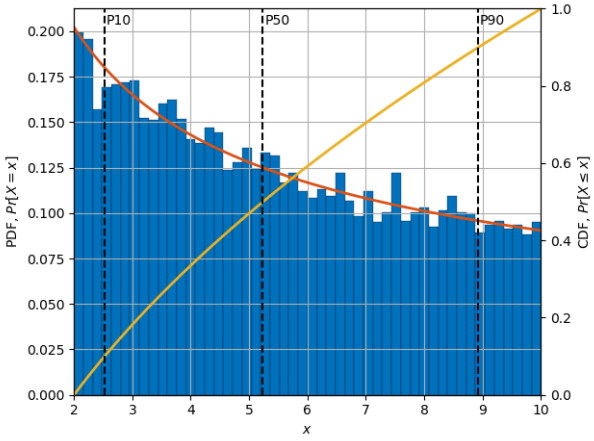
Examples



(a) Linear



(b) Log



(c) Square-root

Figure 5: Examples of variations of the uniform distribution with $a = 2$ and $b = 10$.

4.4.4. Triangular Distribution

Linear Transform

The PDF of the triangular distribution is given by

$$f(x; a, b, c) = \begin{cases} 0, & \text{for } x \leq a \\ \frac{2(x-a)}{(b-a)(c-a)}, & \text{for } a < x \leq c \\ \frac{2(b-x)}{(b-a)(b-c)}, & \text{for } c < x \leq b \\ 0, & \text{for } x > b \end{cases}$$

and the CDF

$$F(x; a, b, c) = \begin{cases} 0, & \text{for } x \leq a \\ \frac{(x-a)^2}{(b-a)(c-a)}, & \text{for } a < x \leq c \\ 1 - \frac{(b-x)^2}{(b-a)(b-c)}, & \text{for } c < x < b \\ 1, & \text{for } x \geq b \end{cases}$$

the inverse cumulative distribution function for mapping between a uniform distribution and a uniform distribution is given by

$$x = \begin{cases} a + \sqrt{y(b-a)(c-a)}, & \text{for } 0 < y < F(c) \\ b - \sqrt{(1-y)(b-a)(b-c)}, & \text{for } F(c) \leq y < 1 \end{cases}$$

where $F(c) = \frac{c-a}{b-a}$.

Log Transform

A variable, X, is said to follow a log-triangular distribution if

$$X = e^Z, \quad Z \sim T(\ln(a), \ln(c), \ln(b))$$

where n is the logarithmic base. Similar to the log-uniform distribution it can be shown that the log-triangular distribution is unaffected by the choice of logarithmic base, n, see Appendix 14.7 for proof. Therefore it is assumed that the logarithmic base corresponds to the natural logarithm, as this simplifies the expressions. The triangular distribution shares the properties of the uniform distribution, namely that the defining parameters are invariant to a monotonic transform. The PDF of the log-triangular distribution is given by

$$f_X(x; a, b, c) = f_Z\left(\ln(x); \ln(a), \ln(b), \ln(c)\right) \cdot \frac{1}{x} = \begin{cases} 0, & \text{for } x \leq a \\ \frac{2\ln\left(\frac{x}{a}\right)}{x\ln\left(\frac{b}{a}\right)\ln\left(\frac{c}{a}\right)}, & \text{for } a < x \leq c \\ \frac{2\ln\left(\frac{b}{x}\right)}{x\ln\left(\frac{b}{a}\right)\ln\left(\frac{b}{c}\right)}, & \text{for } c < x \leq b \\ 0, & \text{for } x > b \end{cases}$$

and the CDF

$$F_X(x; a, b, c) = F_Z\left(\ln(x); \ln(a), \ln(b), \ln(c)\right) = \begin{cases} 0, & \text{for } x \leq a \\ \frac{\ln\left(\frac{x}{a}\right)^2}{\ln\left(\frac{b}{a}\right)\ln\left(\frac{c}{a}\right)}, & \text{for } a < x \leq c \\ 1 - \frac{\ln\left(\frac{b}{x}\right)^2}{\ln\left(\frac{b}{a}\right)\ln\left(\frac{b}{c}\right)}, & \text{for } c < x \leq b \\ 1, & \text{for } x > b \end{cases}$$

and the inverse CDF is given by

$$x = F_X^{-1}(y; a, b, c) = e^{F_Z^{-1}(y; a, b, c)} = \begin{cases} ae^{\sqrt{y\ln\left(\frac{b}{a}\right)\ln\left(\frac{c}{a}\right)}}, & \text{for } 0 < y < F(c) \\ be^{-\sqrt{\ln\left(\frac{b}{a}\right)\ln\left(\frac{b}{c}\right)(1-y)}}, & \text{for } F(c) \leq y < 1 \end{cases}$$

where $F(c) = \log_{\frac{b}{a}}\left(\frac{c}{a}\right)$.

Root Transform

A variable, X, is said to follow a root-triangular distribution if

$$X = Z^n, \quad Z \sim T\left(\sqrt[n]{a}, \sqrt[n]{c}, \sqrt[n]{b}\right)$$

where n is the root. The PDF of the root-triangular distribution is given by

$$f_X(x; a, b, c) = f_Z\left(\sqrt[n]{x}; \sqrt[n]{a}, \sqrt[n]{b}, \sqrt[n]{c}\right) \cdot \frac{1}{x} = \begin{cases} 0, & \text{for } x \leq a \\ \frac{2\left(\sqrt[n]{x} - \sqrt[n]{a}\right)}{\left(\sqrt[n]{b} - \sqrt[n]{a}\right)\left(\sqrt[n]{c} - \sqrt[n]{a}\right)}, & \text{for } a < x \leq c \\ \frac{2\left(\sqrt[n]{b} - \sqrt[n]{x}\right)}{\left(\sqrt[n]{b} - \sqrt[n]{a}\right)\left(\sqrt[n]{b} - \sqrt[n]{c}\right)}, & \text{for } c < x \leq b \\ 0, & \text{for } x > b \end{cases}$$

and the CDF

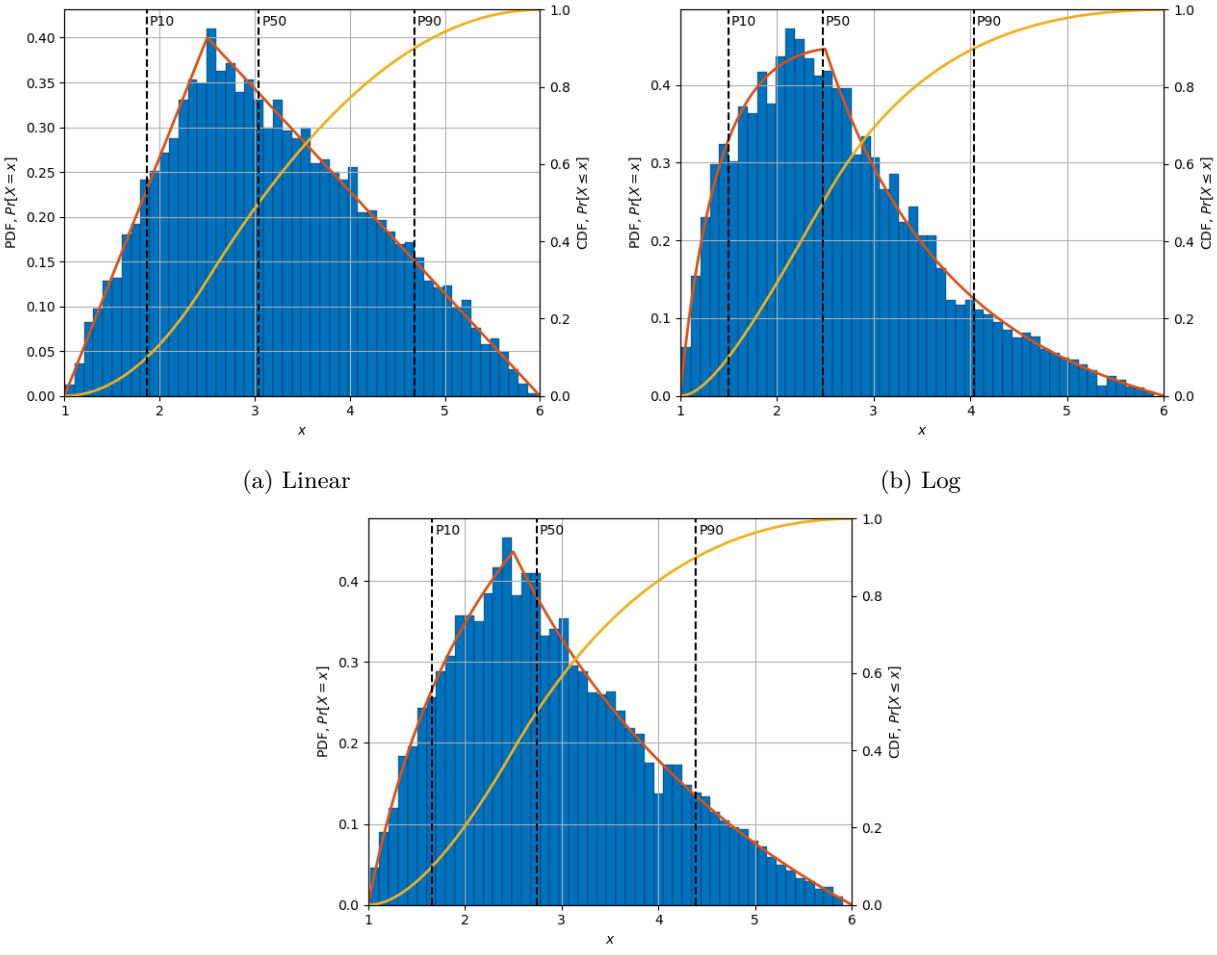
$$F_X(x;a,b,c) = F_Z\left(\sqrt[n]{x};\sqrt[n]{a},\sqrt[n]{b},\sqrt[n]{c}\right) = \begin{cases} 0, & \text{for } x \le a \\ \frac{\left(\sqrt[n]{x}-\sqrt[n]{a}\right)^2}{\left(\sqrt[n]{b}-\sqrt[n]{a}\right)\left(\sqrt[n]{c}-\sqrt[n]{a}\right)}, & \text{for } a < x \le c \\ 1-\frac{\left(\sqrt[n]{b}-\sqrt[n]{x}\right)^2}{\left(\sqrt[n]{b}-\sqrt[n]{a}\right)\left(\sqrt[n]{b}-\sqrt[n]{c}\right)}, & \text{for } c < x \le b \\ 1, & \text{for } x > b \end{cases}$$

and the inverse CDF is given by

$$x = F_X^{-1}(x;a,b,c) = \left(F_Z^{-1}\left(y;\sqrt[n]{a},\sqrt[n]{b},\sqrt[n]{c}\right)\right)^n = \begin{cases} \left(\sqrt[n]{a}+\sqrt{y(\sqrt[n]{b}-\sqrt[n]{a})(\sqrt[n]{c}-\sqrt[n]{a})}\right)^n, & \text{for } 0 < y < F(c) \\ \left(\sqrt[n]{b}-\sqrt{(1-y)(\sqrt[n]{b}-\sqrt[n]{a})(\sqrt[n]{b}-\sqrt[n]{c})}\right)^n, & \text{for } F(c) \le y < 1 \end{cases}$$

where $F(c) = \frac{\sqrt[n]{c}-\sqrt[n]{a}}{\sqrt[n]{b}-\sqrt[n]{a}}$.

Examples



(a) Linear

(b) Log

(c) Square-root

Figure 6: Examples of variations of the triangular distribution with $a = 1$, $b = 6$ and $c = 2.5$.

4.4.5. Exponential Distribution

Linear Transform
The PDF of the exponential distribution is given by

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

where $\lambda > 0$ is a rate parameter. The CDF is given by

$$F(x;\lambda) = \begin{cases} 1 - e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

and the inverse CDF is

$$x = F^{-1}(y;\lambda) = -\frac{\ln(1-p)}{\lambda}, \quad y \in [0,1]$$

4.4.6. Gamma Distribution

Linear Transform
The PDF of the gamma distribution is given by

$$f(x;\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{\beta x}$$

where $\alpha > 0$ is a shape parameter and $\beta > 0$ is rate parameter and $\Gamma(\alpha)$ is the gamma-function. Notice that there exists an alternative formulation with k (shape) and θ (scale) defined by $k = \alpha$ and $\theta = \beta^{-1}$. The CDF is given by

$$F(x;\alpha,\beta) = \frac{1}{\Gamma(\alpha)} \gamma(\beta x;\alpha)$$

where $\gamma(x;\alpha)$ is the lower incomplete gamma function. There exists no closed-form analytical solution to the inverse cumulative distribution function of the gamma distribution. Instead numerical algorithms are used which can sample from the distribution.

4.4.7. Beta Distribution

Linear Transform
The PDF of the beta distribution is given by

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}, \quad x \in [0,1]$$

where $\mathrm{B}(\alpha,\beta)$ is the beta function. The CDF is the regularized incompletely beta function, $I_x(\alpha,\beta)$ defined by

$$F(x;\alpha,\beta) = \frac{\mathrm{B}(x;\alpha,\beta)}{\mathrm{B}(\alpha,\beta)} = I_x(\alpha,\beta), \quad x \in [0,1]$$

with $\mathrm{B}(x;\alpha,\beta)$ being the lower incomplete beta function. There exists no closed-form analytical solution to the inverse cumulative distribution function of the beta distribution. Instead numerical algorithms are used which can sample from the distribution.

The beta distribution can be extended to what is commonly called the 4-parameter beta distribution in order to extend the support from $x \in [0,1]$ to $x \in [a,b]$ by simply normalizing the input variable. Let

$X = Z(b-a) + a$ for $Z \sim B(\alpha, \beta)$ then

$$f_X(x; \alpha, \beta, a, b) = \frac{1}{b-a} \cdot f_Z\left(\frac{x-a}{b-a}; \alpha, \beta\right), \quad x \in [a, b]$$

and similarly for the CDF.

Log Transform
A variable, X, is said to follow a log-beta distribution if

$$X = e^Z, \quad Z \sim B(\alpha, \beta, \ln(a), \ln(b))$$

where n is the logarithmic base. Similar to other distributions it can be shown that the log-beta is invariant to the choice of logarithmic base, n, see Appendix 14.8 for proof. For any monotonic transform of the beta distribution the two shape parameters α and β remain unchanged. For the 4-parameter beta the minimum and maximum are transformed, but are invariant for any monotonic transform. The PDF is

$$f_X(x; \alpha, \beta, a, b) = f_Z\left(\ln(x); \alpha, \beta, \ln(a), \ln(b)\right) \cdot \frac{1}{x} = f_Z\left(\log_{\frac{b}{a}}\left(\frac{x}{a}\right); \alpha, \beta\right) \frac{1}{x \ln\left(\frac{b}{a}\right)}, \quad x \in [a, b]$$

the CDF is

$$F_X(x; \alpha, \beta, a, b) = F_Z\left(\ln(x); \alpha, \beta, \ln(a), \ln(b)\right) = F_Z\left(\log_{\frac{b}{a}}\left(\frac{x}{a}\right); \alpha, \beta\right) = I_z(\alpha, \beta), \quad x \in [a, b]$$

where $z = \log_{\frac{b}{a}}\left(\frac{x}{a}\right)$ and the inverse CDF is

$$F_X^{-1}(y; \alpha, \beta, a, b) = e^{F_Z^{-1}(y; \alpha, \beta, \ln(a), \ln(b))}, \quad y \in [0, 1]$$

Root Transform
A variable, X, is said to follow a root-beta distribution if

$$X = Z^n, \quad Z \sim B\left(\alpha, \beta, \sqrt[n]{a}, \sqrt[n]{b}\right)$$

where n is the exponent. The PDF is

$$f_X(x; \alpha, \beta, a, b) = f_Z(\sqrt[n]{x}; \alpha, \beta, \sqrt[n]{a}, \sqrt[n]{b}) \cdot \frac{x^{\frac{1}{n}-1}}{n} = f_Z\left(\frac{\sqrt[n]{x} - \sqrt[n]{a}}{\sqrt[n]{b} - \sqrt[n]{a}}; \alpha, \beta\right) \cdot \frac{x^{\frac{1}{n}-1}}{n\left(\sqrt[n]{b} - \sqrt[n]{a}\right)}, \quad x \in [a, b]$$

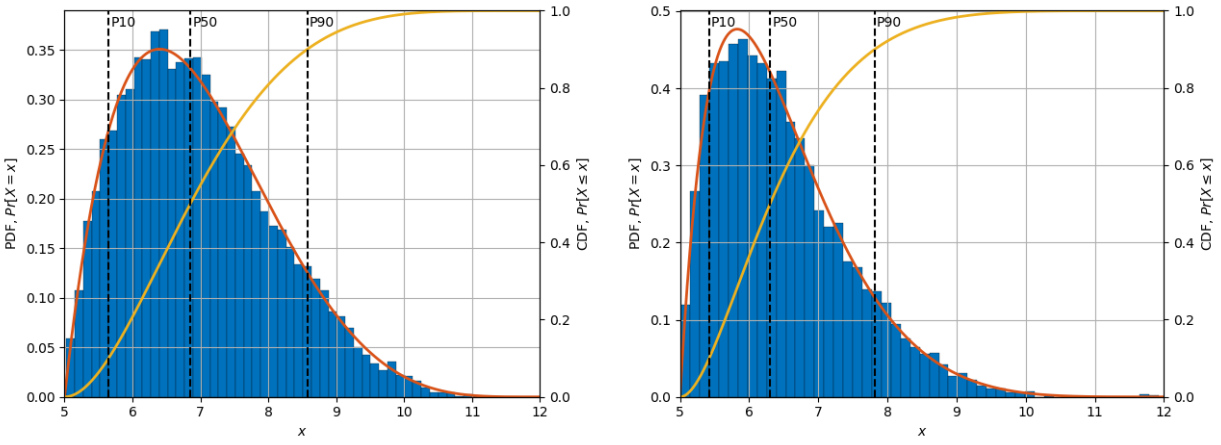
the CDF is

$$F_X(x; \alpha, \beta, a, b) = F_Z(\sqrt[n]{x}; \alpha, \beta, \sqrt[n]{a}, \sqrt[n]{b}) = F_Z\left(\frac{\sqrt[n]{x} - \sqrt[n]{a}}{\sqrt[n]{b} - \sqrt[n]{a}}; \alpha, \beta\right) = I_z(\alpha, \beta), \quad x \in [a, b]$$

where $z = \log_{\frac{b}{a}}\left(\frac{x}{a}\right)$ and the inverse CDF is

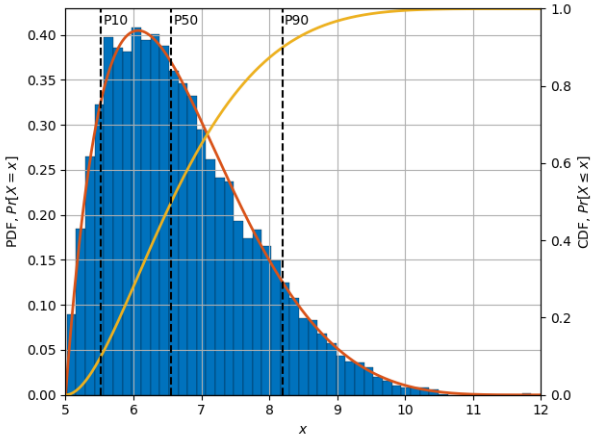
$$F_X^{-1}(y; \alpha, \beta, a, b) = e^{F_Z^{-1}(y; \alpha, \beta, \sqrt[n]{a}, \sqrt[n]{b})}, \quad y \in [0, 1]$$

Examples



(a) Linear

(b) Log



(c) Square-root

Figure 7: Examples of variations of the triangular distribution with $\alpha = 2$, $\beta = 5$, $a = 5$ and $b = 12$.

4.4.8. PERT Distribution

Linear Transform

The PERT distribution is a special case of the 4-parameter beta distribution. Instead of providing the scale parameters α and β, the minimum, maximum and mode are provided. Based on the parameters the scale parameters are determined as follows

$$\alpha = 1 + 4 \cdot \frac{c - a}{b - a}, \quad \beta = 1 + 4 \cdot \frac{b - c}{b - a} \tag{13}$$

The CDF and inverse CDF are same as for the beta.

Log Transform

The calculation of α and β should follow (13) and then follow the log transform of the beta distribution

Root Transform

The calculation of α and β should follow (13) and then follow the root transform of the beta distribution

4.4.9. Discrete Distributions

A discrete distribution requires an arbitrary sequence of outcomes, $(x_1, x_2, ..., x_n)$ and probabilities of these outcomes, $(p_1, p_2, ..., p_n)$ as well as a continuous distribution as defined in Section 4.4. Here it is a requirement

that

$$\sum_{i=1}^{n} p_i = 1$$

Given a uniformly distributed random variable, U then

$$X = \begin{cases} x_1, & \text{for } U \leq p_1 \\ x_2, & \text{for } p_1 < U \leq \sum_{i=1}^{2} p_i \\ \vdots & \\ x_n, & \text{for } \sum_{i=1}^{n-1} p_i < U \leq \sum_{i=1}^{n} p_i \end{cases}$$

The log- and root transform of the discrete distribution is equivalent to replacing the linear uniform distribution, U, but its corresponding transform.

5. Experimental Design

Experimental Design (ED) is the process of designing experiments which provide the maximum amount of information with the minimum number of trials, with a trade-off between the two.

An illustration of an ED in 1D, 2D and 3D can be seen on Figure **??**.

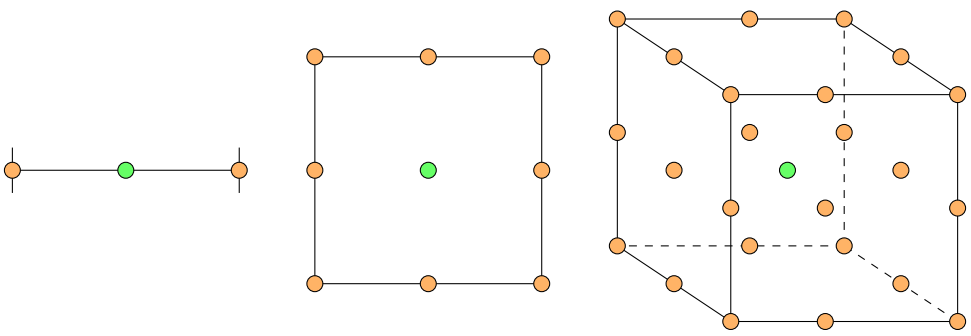


Figure 8: Illustration of a Three-Level Full Factorial Design in 1D, 2D and 3D, using 3^n experiments. Orange are the samples with at least one parameter at min/max and green is the base case.

Example 5.1. Imagine a weight with a certain measurement error, σ. The purpose of a given experiment is to weigh 3 items and determine the weight of each. There are several ways to design an experiment which may solve this problem. First remember that

$$\text{Var}\left[aX - bY\right] = a^2\text{Var}\left[X\right] + b^2\text{Var}\left[Y\right] - 2ab\text{Cov}\left[X, Y\right] = a^2\sigma^2 + b^2\sigma^2 = (a^2 + b^2)\sigma^2 \tag{14}$$

by the assumption that X and Y are independent, thus $\text{Cov}\left[X, Y\right] = 0$. The simplest solution to the problem is weighing each item one at a time

Table 2: Experimental Design 1

Trial	Configuration			
	I_1	I_2	I_3	Result
Trial 1	0	0	0	Y_1
Trial 2	1	0	0	Y_2
Trial 3	0	1	0	Y_3
Trial 4	0	0	1	Y_4

Note that a positive 1 means the item is on the right side of weight and a -1 means the item is on the left side of the weight. For Design 1 the weight of each item is evaluated as $I_1 = Y_2 - Y_1$, $I_2 = Y_3 - Y_1$, $I_3 = Y_4 - Y_1$ which as per (14) has the uncertainty $2\sigma^2$. Consequently the cost of the experiment is 4 trials with a quality of $2\sigma^2$.

Imagine a different type of design

Table 3: Experimental Design 2

Trial	Configuration			
	I_1	I_2	I_3	Result
Trial 1	0	0	0	Y_1
Trial 2	1	1	0	Y_2
Trial 3	1	0	1	Y_3
Trial 4	0	1	1	Y_4

The quality of this experiment is slightly more difficult to calculate. To find the the weight of I_1, the rows have to be solved such that the I_2 and I_3 column are 0, while the I_1 column is > 0, and similarly for calculating the others. Here shown for I_1

$$I_1 = \frac{Y_2 + Y_3 - Y_1 - Y_4}{2} \Leftrightarrow \mathrm{Var}[I_1] = \frac{1}{2^2}\left(\mathrm{Var}[Y_2] + \mathrm{Var}[Y_3] + \mathrm{Var}[Y_1] + \mathrm{Var}[Y_4]\right) = \frac{4\sigma^2}{4} = \sigma^2$$

and similarly for the other items. Consequently, the cost of the experiment is 4 trials with a quality of σ^2. This is the same amount of trials as Experiment Design 1, but half the uncertainty, so clearly the design of an experiment has an influence on the quality of the measurements. So far only the right hand-side of the weight has been used in the experiment, and consequently results compared relative to Trial 1 which is empty. By introducing counterweight measurements, the design can be optimized even further

Table 4: Experimental Design 3

Trial	Configuration			
	I_1	I_2	I_3	Result
Trial 1	1	1	1	Y_1
Trial 2	1	-1	-1	Y_2
Trial 3	-1	1	-1	Y_3
Trial 4	-1	-1	1	Y_4

The quality of this experiment has to be calculated as follows

$$I_1 = \frac{Y_3 + Y_4 - Y_1 - Y_2}{-4} \Leftrightarrow \text{Var}[I_1] = \frac{1}{(-4)^2}\left(\text{Var}[Y_2] + \text{Var}[Y_3] + \text{Var}[Y_1] + \text{Var}[Y_4]\right) = \frac{4\sigma^2}{16} = \frac{1}{4}\sigma^2$$

For this design the cost is again 4 trials, but the quality of the measurements are $0.25 \cdot \sigma^2$ i.e. 8 times better than Design 1.

As shown in Example 5.1 the design of an experiment can be optimized to yield additional information with the same amount of trials. Obviously, the problem described in Example 5.1 is very simple and consequently finding an optimal design by hand is possible. However, as complexity grows it becomes infeasible to create designs manually, which is where design algorithms come in. There exists multiple different types of algorithms, but they can broadly be categorized in two different classes namely

- Factorial Design (FD)
- Space Filling Design (SFD)

where an FD samples the extrema of a unit hypercube, an SFD attempts to fill the space of the unit hypercube.

5.1. Factorial Design

A factorial design provides a consistent approach to maximize the information, based on a given optimality condition. When the design problem is assumed to be linear, i.e. of the form

$$Y = X\beta + r$$

where Y is the modelled response, X is the design of the experiment, β is the impact and r is the residuals. Then a design can be optimized on the basis of different optimality conditions.

A-optimability:
A-optimality (average) strives to minimize the trace of the inverse of the Fisher's information matrix. This means it minimizes the average variance of the estimated regressions coefficients.

C-optimality:
C-optimality seeks to minimize the variance of a BLUE (best linear unbiased estimator) of a predetermined linear combination of model parameters.

D-optimality:
D-optimality (determinant) attempts to minimize $||(X^TX)^{-1}||$, i.e. the determinant of the inverse of Fisher's information matrix or conversely maximize the determinant of Fisher's information matrix X^TX. This is the same as maximizing the differential Shannon information content of the parameter estimates.

E-optimality:
E-optimality (eigenvalue) instead maximizes the minimum eigenvalue of Fisher's information matrix.

T-optimality:
T-optimality strives to maximize the trace of Fisher's information matrix.

G-optimality:
G-optimality is concerned with the variance of the predictions and seeks to minimize the maximum entry of the diagonal of the projection matrix $X(X^TX)^{-1}X^T$. This has the effect of minimizing the maximum variance of the predicted values.

I-optimality:
I-optimality (integrated) also strives to reduce the variance of the predictions, which is done by minimizing the average prediction variance over the design space.

V-optimality:
V-optimality (variance) is the third criterion reducing the variance of the predictions. This seeks to minimize the average prediction variance over a set of m specific points.

5.1.1. Mono-Sensitivity

Mono-sensitivity is a method for quantifying the impact of individual parameters. One parameter is varied to its minimum and maximum respectively while all other parameters are kept at their most likely. This is done one parameter at a time and the impact is compared to a reference case where all parameters are kept at their most likely. This method requires $2n + 1$ trials where n is the number of parameters.

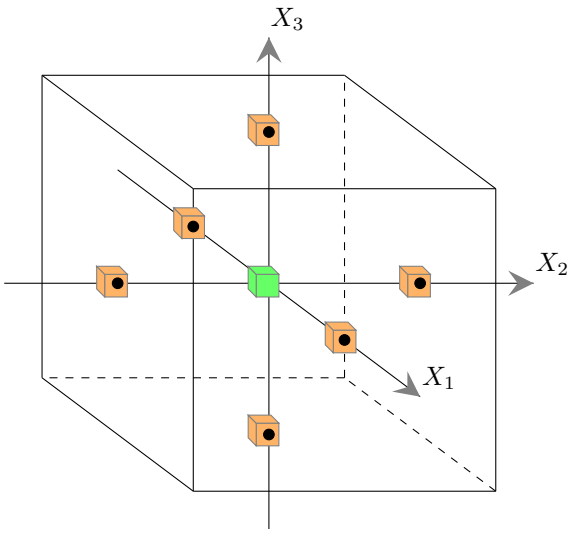


Figure 9: Mono-Sensitivity: Unit hypercube with 3 parameters. Arrows shows the axis' and black dots the min/max per variable. Cubes denote the sampling points, orange are for min/max variation and green for the reference point.

A Mono-sensitivity design allows for exactly quantifying the impact of a single parameter, but does not take into account interactions between parameters. The purpose of a mono-sensitivity study is typically to generate a tornado diagram. That is, evaluate the impact of each parameter in the positive and negative direction relative to the reference point. I.e.

$$a_{1+} = Y_{1+} - Y_{ref}, \quad a_{1-} = Y_{1-} - Y_{ref}$$

and similarly for the variable X_2 and X_3. Here the total impact of variable X_1 is given by $a_{1+} + a_{1-}$.

5.1.2. Full Factorial Design

The full factorial design is a method for quantifying the impact of individual parameters as well as the interactions between them. It is sampled at all extrema combinations of all variables, hence it requires 2^n runs this quickly becoming infeasible in practice when $n > 6$.

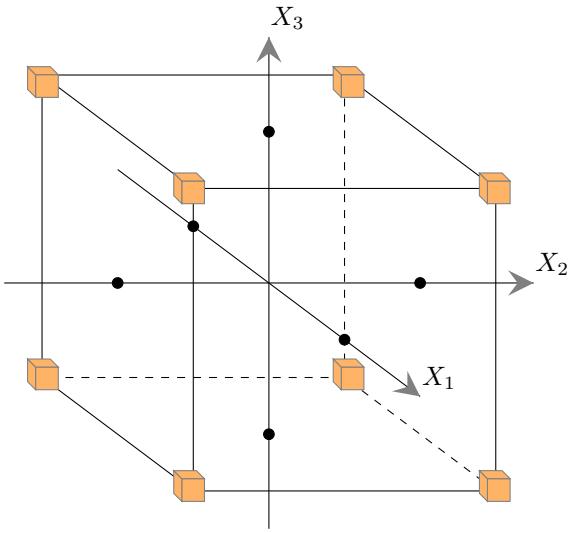


Figure 10: Full Factorial: Unit hypercube with 3 parameters. Arrows shows the axis' and black dots the min/max per variable. Cubes denote the sampling points.

The design assumes the impact of the parameters on the solution is linear between the minimum and maximum. This is of course a naive assumption to make, but can be acceptable dependent on the problem at hand and purpose of the design. It does however allow for including interactions terms in the linear expression.

$$Y = a_0 + a_1X_1 + a_2X_2 + a_3X_3 + a_{12}X_1X_2 + a_{13}X_1X_3 + a_{23}X_2X_3 + a_{123}X_1X_2X_3 \tag{15}$$

which with 2^n results and 2^n unknowns has a unique solution found by solving a linear system.

5.1.3. Fractional Factorial Design

The fractional factorial design is a reduced version of the full factorial design. As can be seen from (15) a 3^{rd}-order term is included and as n increases the number of higher order terms will increase. While interactions between two parameters can have a significant impact, the impact of higher order terms are typically negligible. This means terms can be eliminated from the equation and consequently less results are required to solve the system of equations.

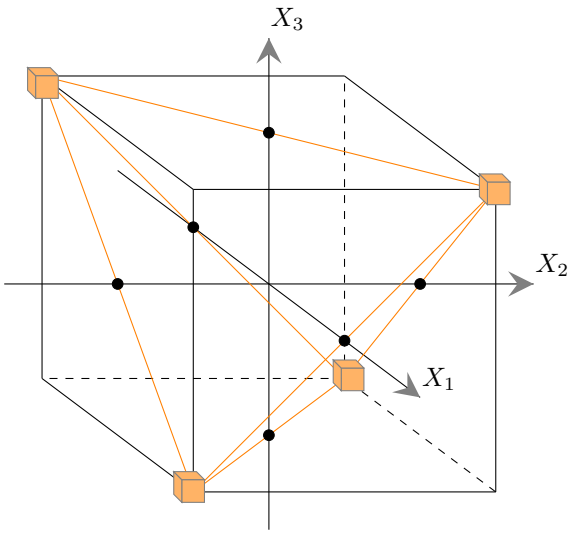


Figure 11: Fractional Factorial: Unit hypercube with 3 parameters. Arrows shows the axis' and black dots the min/max per variable. Cubes denote the sampling points.

A fractional factorial design requires 2^{n-p} runs whereas the full factorial design requires 2^n. The logic here is to avoid higher order interaction terms when n is large. In the specific case shown on Fig. 12 with $n = 3$ it poses a problem as only 4 results are available and there are a total of 7 interesting coefficients. In general when only the 0^{th}-, 1^{st}- and 2^{nd}-order terms are of interest, then the number of required coefficients becomes $1 + \frac{1}{2}n + \frac{1}{2}n^2$. From this it is evident that for the maximum information fractional factorial design, i.e. with $p = 1$, it requires $n \geq 5$ in order to use the fractional design and be able to estimate all coefficients of interest. As n becomes larger, it is also possible to increase p. The value $n - p$ is called the resolution of the design. This gives rise to the term *aliasing*. Aliasing is the process of combining coefficients such that they describe multiple interactions. For the 3 parameter problem seen on Fig. 12 the aliasing scheme with $p = 1$ would be a Resolution II design

$$l_0 = a_0 + a_{123}$$
$$l_1 = a_1 + a_{23}$$
$$l_2 = a_2 + a_{13}$$
$$l_3 = a_3 + a_{12}$$

where l_i is called a contrast. As mentioned when $n < 5$ aliasing becomes an issue as coefficients of interest becomes mixed. As n increases it becomes possible to mix lower order coefficients (up to and including 2^{nd}-order) with higher order coefficients which are assumed to have a low impact.

5.1.4. Modified Fractional Factorial Design

No reference available, idea by the author. As per the example with weights there is a chance this is an inferior design, and more information is in fact capture when sampling at the nodes of the cube.

The modified fractional factorial design is an attempt to deal with the issue of aliasing. Given that only terms up to and including the interaction between two variables at a time are of interest, it must be possible to provide a design which is optimized for this.

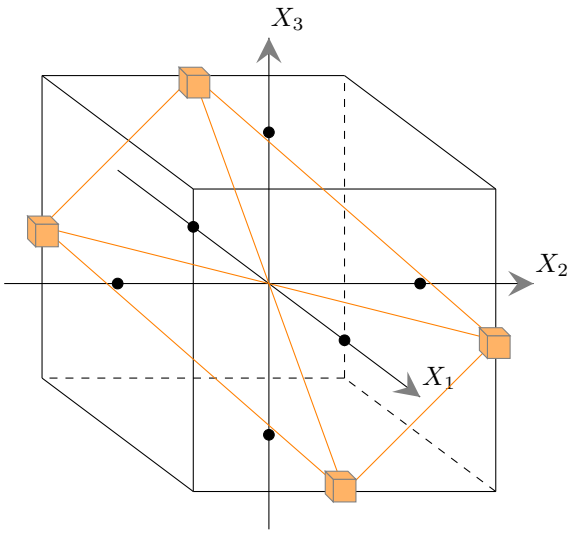


Figure 12: Modified Fractional Factorial: Unit hypercube with 3 parameters. Arrows shows the axis' and black dots the min/max per variable. Cubes denote the sampling points.

Because the modified fractional factorial design only samples at the extrema of two variables at a time, it is easier to isolate the 2^{nd}-order interactions without undue influence from the additional parameters. Whether

the reduced sampling at end-points increases the uncertainty on the 1st-order coefficients, will have to be investigated further.

5.1.5. Central Composite Design

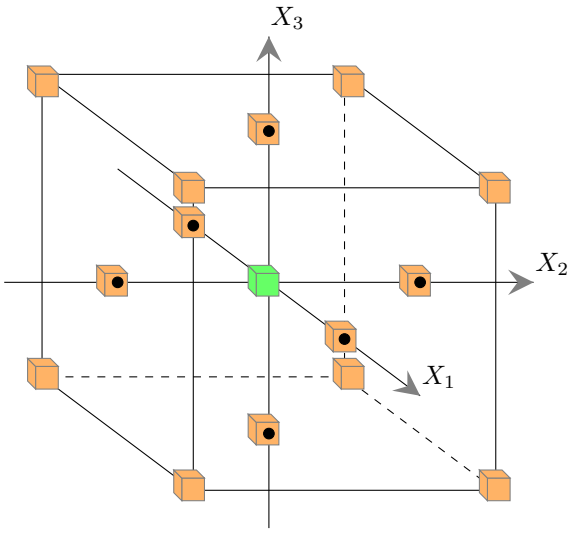


Figure 13: Central Composite Design: Unit hypercube with 3 parameters. Arrows shows the axis' and black dots the min/max per variable. Cubes denote the sampling points, orange are for min/max variation and green for the reference point.

5.1.6. Box-Behnken Design

5.2. Space Filling Design

5.2.1. Latin Hypercube

5.2.2. Kennard-Stone

5.2.3. WSP

The WSP algorithm is an SFD that was originally invented by Wootton, Sergent, Phan-Tan-Luu from whom it derives its name.

6. Objective Functions

The objective functions considered here are in the scope of regressions analysis, i.e. objective functions used to fit parameters to a model.

6.1. Least-Squares Estimation

Least-Squares estimation (LS) is a method which minimizes the sum of squared distances between the data and the model. It is applicable to linear models of the form

$$y_i = f(x_i, \beta) + r_i = x_i\beta + r_i, \quad r_i \sim \mathcal{N}\left(0, \sigma_i^2\right)$$

where y_i is the data $f(\cdot)$ is the model, x_i is the independent variable, β is the model parameters and r_i is the residual. The least squares formulation is given by

$$OF(\beta) = \sum_{i=1}^{n} \omega_i r_i \left(\beta\right)^2 = \sum_{i=1}^{n} \omega_i \left(y_i - f(x_i, \beta)\right)^2$$

where OF is the objective function and ω_i is a weight. For the linear case this can be solved by differentiating and solving for β by setting the derivative equal to zero. LS is considered BLUE (best linear unbiased estimator) if the model is linear and the residuals are uncorrelated with equal variance and zero mean. This can be achieved by setting $\omega_i = \frac{1}{\sigma_i^2}$. In fact for LS to be BLUE it is not required that the residuals are normally distributed, nor that they are identical or independent (only uncorrelated and homoscedastic).

For any model which can be written on an algebraic form

$$y = X\beta + r, \quad r \in \mathcal{N}(0,\Sigma)$$

an estimate, $\hat{\beta}$ is given by the generalized least squares estimator

$$\hat{\beta} = \left(X^T \Sigma^{-1} X\right)^{-1} X^T \Sigma^{-1} y$$

which can also be formulated as weighted least squares

$$\hat{\beta} = \left(X^T W X\right)^{-1} X^T W y$$

allowing $W \neq \Sigma^{-1}$ or in the form of ordinary least squares where $W = I$

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T y \tag{16}$$

6.2. Likelihood Estimation

The likelihood function measures the goodness of fit of a statistical model to a sample of data for given values of an unknown parameter, θ. The likelihood function can be written as

$$\mathcal{L}(\theta \mid x) = f(x \mid \theta) \tag{17}$$

where x is the outcome of an experiment and θ is the parameters. This means that $f(x \mid \theta)$ viewed as a function of x with θ fixed is a probability distribution, but when viewed as a function of θ with x fixed is a likelihood function. It is important to note here that the likelihood is not a PDF, as the integral of $\mathcal{L}(\theta \mid x)$ over θ does not integrate to unity. The likelihood in and of itself does not describe the quality of a parameter fit, but rather the relative likelihood between different parameter fits describes the goodness of one fit over the over

$$\Lambda(\theta_1, \theta_2 \mid x) = \frac{\mathcal{L}(\theta_1 \mid x)}{\mathcal{L}(\theta_2 \mid x)}$$

with $\Lambda > 1$ denoting that θ_1 is more likely than θ_2 given the information, x.

When multiple data is available, the likelihood function follows the joint probability density function. Under the assumption that the uncertainty of the individual data points are independent, then the joint PDF is the product of the marginal PDFs

$$f_{X,Y}(x,y) = f_X(x) f_Y(y) \tag{18}$$

given that (18) quickly diminishes towards 0, which is impractical for numerical algorithms, the logarithm of the likelihood function - also called the log-likelihood - is typically considered. For n data points the

log-likelihood is given as

$$\mathcal{L}(\theta \mid x) = \prod_{i=1}^{n} f(x \mid \theta) \Leftrightarrow \ln\left(\mathcal{L}(\theta \mid x)\right) = \ell(\theta \mid x) = \ln\left(\prod_{i=1}^{n} f(x \mid \theta)\right) = \sum_{i=1}^{n} \ln(f(x \mid \theta))$$

with the log-likelihood often denoted by $\ell(\theta \mid x)$. Since the logarithmic transform is monotone the properties w.r.t. to relative likelihood is preserved. Notice that maximizing the likelihood is equivalent to minimizing the negative log-likelihood, i.e.

$$\max_{\theta} \mathcal{L}(\theta \mid x) \Leftrightarrow \min_{\theta} -\ell(\theta \mid x)$$

7. Proxy Models

A proxy model (response surface, meta-model, etc.) is - in the context of `modpy` - a model fitted to a set of simulated points. Let $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)^T$ be a set of sampled points and $Y = (f_{\text{sim}}(\boldsymbol{x}_1), ..., f_{\text{sim}}(\boldsymbol{x}_n))^T$ be the corresponding simulation results. Then the proxy model, $M(\boldsymbol{x})$ approximates y at point $\boldsymbol{x}$, that is

$$y = f_{\text{sim}}(\boldsymbol{x}), \quad \hat{y} = M(\boldsymbol{x})$$

I.e. $M(\cdot)$ is an estimator of $f_{\text{sim}}(\cdot)$. Generally proxy models allows for both interpolation and extrapolation. The primary purpose in `modpy` is to explore an uncertain domain, $X \in \Omega \subset \mathbb{R}^d$ by means of an optimization algorithm. Consequently, the choice of proxy model is based on interpolation properties rather than extrapolation.

7.1. Polynomial Models

7.1.1. Linear Model

Besides a single constant, a linear model is the simplest polynomial model that can be fitted to data. For a simple bivariate case the model is given by

$$f(x_1, x_2) = a_0 + a_1 x_1 + a_2 x_2 + a_{12} x_1 x_2$$

when interactions are included the number of terms quickly explode to infeasible numbers as the number of variables, n, increases. For this reason the same assumption as for the Experimental Designs is made, namely that only terms up to and including 2^{nd}-order will be included which for a polynomial with n variables can be written in compact form as

$$f(x_1, x_2, ..., x_n) = a_0 + \sum_{i=1}^{n} \left(a_i x_i + \sum_{j=i+1}^{n} a_{ij} x_i x_j \right)$$

Given a response vector Y $(k \times 1)$ and a corresponding series of points $X = (X_1, X_2, ..., X_n)$ this can be formulated in an algebraic form and solved using (16)

$$Y = X\beta \Leftrightarrow \begin{bmatrix} 1 & x_{1,1} & x_{2,1} & \dots & x_{n,1} & x_{1,1}x_{2,1} & \dots & x_{1,1}x_{n,1} & \dots & x_{n-1,1}x_{n,1} \\ 1 & x_{1,2} & x_{2,2} & \dots & x_{n,2} & x_{1,2}x_{2,2} & \dots & x_{1,2}x_{n,2} & \dots & x_{n-1,2}x_{n,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{1,k} & x_{2,k} & \dots & x_{n,k} & x_{1,k}x_{2,k} & \dots & x_{1,k}x_{n,k} & \dots & x_{n-1,k}x_{n,k} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \\ a_{12} \\ \vdots \\ a_{1n} \\ \vdots \\ a_{(n-1)n} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} \tag{19}$$

7.1.2. Quadratic Model

Often linear models are too simplistic and require non-linear terms in order to appropriately model complex responses. The simplest non-linear model is the quadratic model. For the bivariate case this is written as

$$f(x_1, x_2) = a_0 + a_1x_1 + a_2x_2 + a_{11}x_1^2 + a_{22}x_2^2 + a_{12}x_1x_2$$

the full quadratic model actually includes three additional terms, namely $x_1^2x_2$, $x_1x_2^2$ and $x_1^2x_2^2$. While these terms are 2-way interactions they are higher than 2^{nd}-order (sum of exponents), and are consequently ignored. The model can again be written in compact form for an arbitrary number of variables, n

$$f(x_1, x_2, ..., x_n) = a_0 + \sum_{i=1}^{n} \left(a_ix_i + a_{ii}x_i^2 + \sum_{j=i+1}^{n} a_{ij}x_ix_j \right)$$

It is important to note that the quadratic model is non-linear in the variables, not the parameters and consequently they can be optimized using a linear formulation similar to (19)

$$Y = X\beta \Leftrightarrow \begin{bmatrix} 1 & x_{1,1} & \dots & x_{n,1} & x_{1,1}^2 & \dots & x_{n,1}^2 & x_{1,1}x_{2,1} & \dots & x_{n-1,1}x_{n,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{1,k} & \dots & x_{n,k} & x_{1,k}^2 & \dots & x_{1,k}^2 & x_{1,k}x_{2,k} & \dots & x_{n-1,k}x_{n,k} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \\ a_{11} \\ \vdots \\ a_{nn} \\ a_{12} \\ \vdots \\ a_{(n-1)n} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} \tag{20}$$

which includes all the same terms as (19), along with the n additional quadratic terms.

7.1.3. Cubic Model

In case responses are highly non-linear a quadratic model may not be sufficient to capture it. In this case a cubic model can be utilized by introducing even higher order terms. In the bivariate case

$$f(x_1, x_2) = a_0 + a_1x_1 + a_2x_2 + a_{11}x_1^2 + a_{22}x_2^2 + a_{111}x_1^3 + a_{222}x_2^3 + a_{12}x_1x_2$$

similar to the quadratic model this model could include additional terms, both quadratic and cubic interactions, which are ignored for the same reason. The model is written in compact form as

$$f(x_1, x_2, ..., x_n) = a_0 + \sum_{i=1}^{n} \left(a_i x_i + a_{ii} x_i^2 + a_{iii} x_i^3 + \sum_{j=i+1}^{n} a_{ij} x_i x_j \right)$$

similar to the linear and quadratic model this can be written in terms of linear algebra

$$Y = X\beta \Leftrightarrow \begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,1}^2 & \dots & x_{1,1}^3 & \dots & x_{1,1}x_{2,1} & \dots & x_{n-1,1}x_{n,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{1,k} & \dots & x_{1,k}^2 & \dots & x_{1,k}^3 & \dots & x_{1,k}x_{2,k} & \dots & x_{n-1,k}x_{n,k} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_{11} \\ \vdots \\ a_{111} \\ \vdots \\ a_{12} \\ \vdots \\ a_{(n-1)n} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix}$$

where the system matrix and parameter vector contains all the same terms as (20) with n additional cubic terms.

7.1.4. Model Reduction Techniques

Even when keeping only 2$^{\text{nd}}$-order interaction terms, the number of coefficients can quickly grow as n increases. For a linear, quadratic and cubic respectively

$$\begin{aligned} \text{Linear:} \quad & 1 + \frac{1}{2}n + \frac{1}{2}n^2 \\ \text{Quadratic:} \quad & 1 + \frac{3}{2}n + \frac{1}{2}n^2 \\ \text{Cubic:} \quad & 1 + \frac{5}{2}n + \frac{1}{2}n^2 \end{aligned}$$

In order to avoid over-parameterization model reduction techniques should be used to remove any unnecessary coefficients. This is an iterative process where the full model is fitted, then a tests for statistical significance of the parameters is made and the most insignificant parameter removed. The model re-fitted and a new statistical test made. This continuous until only significant parameters remain.

7.2. Splines

7.3. Kriging

Kriging assumes the true underlying model follows a Gaussian Process

$$Z(\boldsymbol{x}) = \mu(\boldsymbol{x}; \boldsymbol{\beta}) + \sigma^2 Y(\boldsymbol{x}; \boldsymbol{\theta}) \tag{21}$$

where $\mu(\boldsymbol{x}; \boldsymbol{\beta}) \in \mathbb{R}$ is the mean (also called the *trend*), $\sigma \in \mathbb{R}_+$ is the standard deviation and $Y(\boldsymbol{x}; \boldsymbol{\theta}) \sim \mathcal{N}(0, \boldsymbol{R})$ is a Gaussian Process with zero-mean and unit variance.

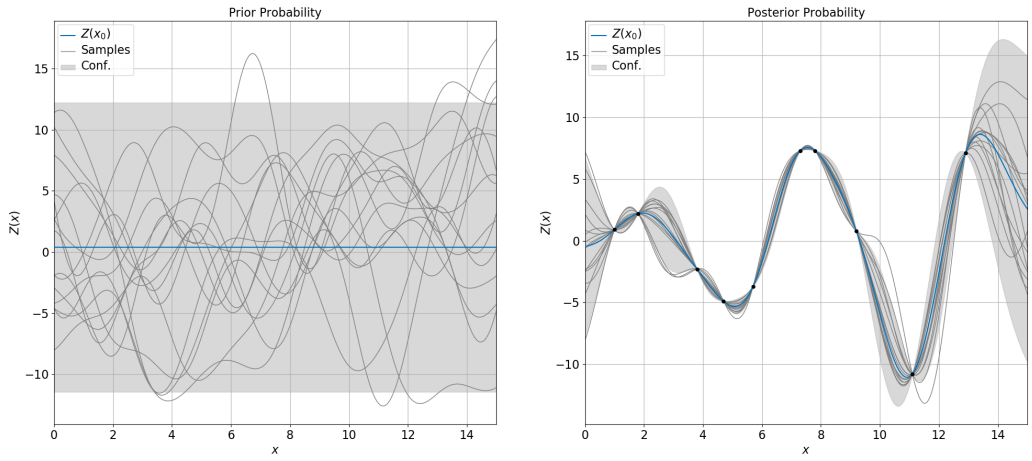


Figure 14: Illustration of a *prior* Gaussian Process (left) and *posterior* Gaussian Process (right). The blue line is the mean, the shaded area is the 95 % confidence interval and the grey lines are randomly sampled processes.

$Y(\boldsymbol{x}, \boldsymbol{\theta})$ can be fully characterised by the auto-correlation function between two sample points, $R(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{\theta})$. Kriging is a statistical method for interpolating between known sample points. It originated as a geostatistical method in the mining-industry and is based on the simple logic that physical properties in a geographically close region are more likely to be similar than the same property further away. In a purely mathematical sense physical properties would mean estimates of any variety and geographically close corresponds to an arbitrary measure of distance.

Kriging has also become popular as a proxy model for computationally expensive simulations. While the mathematics for the two problems - geostatics and proxy modelling - are similar, the approach used often differs. In geostatistics the underlying variables are often the 2D horizontal plane and has similar length scales. In proxy modelling the variables are often in the tens or hundreds and can vary by several orders of magnitude in scale. This leads to different approaches to estimating the covariance function. Only the proxy model approach is introduced here, for a review of the geostatistical one see Appendix 15.

As an interpolator kriging has two desirable properties. It is an exact interpolator, meaning it will exactly honor the simulated point. Secondly, it can estimate the prediction variance at any given point, yielding a (probable) bound on the error of the estimate.

There exists many types of kriging, but the most common ones are

- Simple Kriging
- Ordinary Kriging
- Universal Kriging

they all follow the assumption of (21), but differ in how they model the trend. Kriging is a linear estimator

$$Z^*_{\boldsymbol{\omega}}(\boldsymbol{x}_0) = \sum_{i=1}^{n} \omega_i Z(\boldsymbol{x}_i) = \boldsymbol{\omega}^T \boldsymbol{Z}$$

for Z at point $\boldsymbol{x}_0$ where the weights, $\omega_i \in \mathbb{R}$, are chosen such that they minimize the variance of the predictions. See [11] for further details. If the weights are chosen appropriately then $Z^*_{\boldsymbol{\omega}}$ constitutes a Best Linear Unbiased

Prediction (BLUP) for the interpolated point. The choice of weights depends on the covariance function

$$C(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}) = \sigma^2 R(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}), \quad (i,j) = 1, ..., n$$

is also called the *kernel* and can be inferred from the available simulations.

7.3.1. Trend Function

As seen from (21) the trend, $\mu(\boldsymbol{x}; \boldsymbol{\beta})$, is a function of $\boldsymbol{x}$ and is written as a linear combination

$$\mu(\boldsymbol{x}; \boldsymbol{\beta}) = \sum_{l=0}^{L} f_l(\boldsymbol{x}) \beta_l = \boldsymbol{f}(\boldsymbol{x})^T \boldsymbol{\beta}$$

by convention $f_0(\boldsymbol{x}) = 1$ is a constant with the remainder, $l = 1, ..., L$ being arbitrary functions of $\boldsymbol{x}$. Here $\boldsymbol{\beta} = (\beta_1, ..., \beta_L)^T$ is a set of $L + 1$ unknown hyper-parameters to be tuned. Typical options of $f(\boldsymbol{x})$ are linear and quadratic, i.e.

$$\begin{aligned} \text{Linear:} \quad & \beta_0 + \sum_{l=1}^{L} \beta_l x_l \\ \text{Quadratic:} \quad & \beta_0 + \sum_{l=1}^{L} \beta_l x_l + \sum_{i=1}^{L} \sum_{j=1}^{L} \beta_{ij} x_i x_j \end{aligned}$$

7.3.2. Kernel Function

The kernel function is characterised by the auto-correlation function, R and is a measure of similarity between points. The closer two points are the higher the covariance. To satisfy the assumptions of kriging, a kernel function has to satisfy some strict properties. For this reason a suite of available functions is typically used. In all cases the kernel function is given by

$$C(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma^2 \prod_{k=1}^{d} R(|x_{i,k} - x_{j,k}|)$$

where $R(\cdot,\cdot)$ is a correlation function defined as

$$\begin{aligned} \text{Exponential:} \quad & R(h;\theta) = e^{-\frac{|h|}{|\theta|}} \\ \text{Gaussian:} \quad & R(h;\theta) = e^{-\frac{1}{2}\left(\frac{h}{\theta}\right)^2} \\ \text{Matérn } \nu = \frac{3}{2}\text{:} \quad & R(h;\theta) = \left(1 + \frac{\sqrt{3}|h|}{\theta}\right) e^{-\frac{\sqrt{3}|h|}{\theta}} \\ \text{Matérn } \nu = \frac{5}{2}\text{:} \quad & R(h;\theta) = \left(1 + \frac{\sqrt{5}|h|}{\theta} + \frac{5h^2}{3\theta^2}\right) e^{-\frac{\sqrt{5}|h|}{\theta}} \\ \text{Power-Exponential:} \quad & R(h;\theta) = e^{-\left(\frac{|h|}{\theta}\right)^p}, \quad 0 < p \leq 2 \end{aligned}$$

the parameters, σ and $\boldsymbol{\theta} = (\theta_1, ..., \theta_d)^T$ are called hyper-parameters and typically constitutes a set of $d + 1$ unknown hyper-parameters to be tuned.

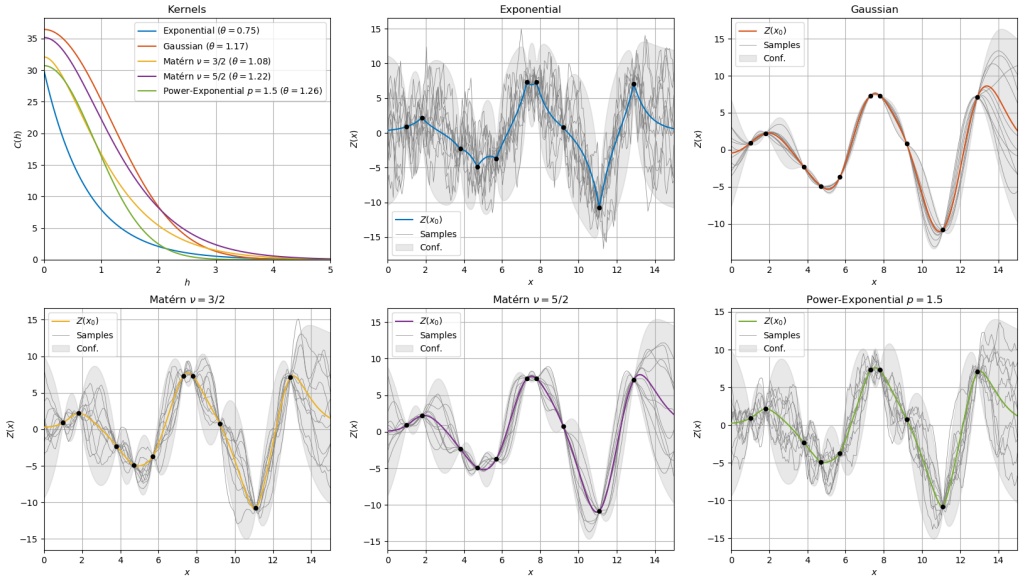


Figure 15: The five different kernel functions fitted to a set of 1D data. The hyper-paramerers are fitted using Maximum Likelihood. Top left figure shows the fitted kernel functions. The remaining figures shows how the respective kernels models the set of 1D data. The colored lines are the mean prediction, the shaded area the 95 %-confidence interval and the gray lines are random samples from a Gaussian Process using the posterior probability.

7.3.3. Hyper-Parameter Estimation

Several methods exists for tuning the set of $L + d + 2$ hyper parameters, $\boldsymbol{\beta}$, σ^2 and $\boldsymbol{\theta}$. The two most common are Maximum Likelihood Estimation and Cross-Validation.

Maximum Likelihood Estimation

Maximum Likelihood Estimation follows the theory outlined in Section 6.2. Given the underlying assumption of a Gaussian Process, the likelihood of observing the given simulation result as a function of the hyper-parameters is a multi-variate normal distribution

$$\mathcal{L}(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta} \mid \boldsymbol{z}) = p(\boldsymbol{z} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{C})}} e^{-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu})\boldsymbol{C}^{-1}(\boldsymbol{z}-\boldsymbol{\mu})}$$

where $\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{x}; \boldsymbol{\beta})$ and $\boldsymbol{C} = \boldsymbol{C}(\boldsymbol{x}, \boldsymbol{x}; \sigma^2, \boldsymbol{\theta})$. The objective function of this problem can be formulated as

$$-\ell(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta} \mid \boldsymbol{z}) = \frac{1}{2}\ln(\det(\boldsymbol{R})) + \frac{n}{2}\ln(2\pi\sigma^2) + \frac{n}{2} \tag{22}$$

For any given value of $\boldsymbol{\theta}$, the maximisation of the likelihood w.r.t. $\boldsymbol{\beta}$ and σ^2 constitutes a convex quadratic programming [12]. Consequently, there exists closed form solution for both. For $\boldsymbol{\beta}$ it is a weighted least-squares estimate

$$\beta = \beta(\boldsymbol{\theta}) = \left(\boldsymbol{F}^T \boldsymbol{R}^{-1} \boldsymbol{F}\right)^{-1} \boldsymbol{F}^T \boldsymbol{R}^{-1} \boldsymbol{z}$$

and for σ^2 it is directly available from differentiating the objective function w.r.t. σ and solving it equal to zero

$$\sigma^2 = \sigma^2(\boldsymbol{\theta}) = \frac{1}{n}(\boldsymbol{z} - \boldsymbol{F}\boldsymbol{\beta})^T \boldsymbol{R}^{-1} (\boldsymbol{z} - \boldsymbol{F}\boldsymbol{\beta})$$

There exists no closed-form solution for estimating $\boldsymbol{\theta}$ and thus it has to be done numerically by minimizing

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & -\ell(\boldsymbol{\theta} \mid \boldsymbol{z}) \\ \text{s.t.} \quad & \boldsymbol{\theta} > \mathbf{0} \end{aligned}$$

This problem is non-linear and multi-modal. Further to this, it is a computationally expensive problem due to the multiple inversions of $\boldsymbol{R}$. Consequenrly, the problem is best solved by a so-called *hybrid* algorithm which is a mix of an evolution algorithm (global) and a gradient-based (local) algorithm.

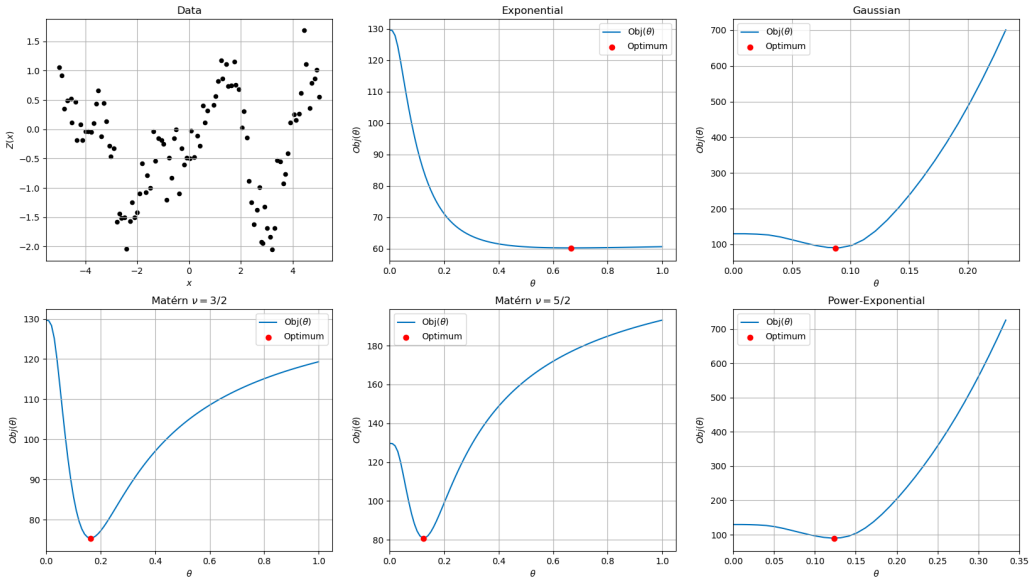


Figure 16: Illustration of the objective function and the optimum for the five possible correlation functions for a 1D ordinary kriging model. Top left shows the data the model is being fitted to.

As seen from Fig. 16 the objective function is non-linear. When the problem is one-dimensional it is not multi-modal, but other complexities arise. The parameter in Fig. 16 was evaluated for $\theta \in [0, 1]$, but as can be seen from the illustration of the gaussian and power-exponential function they both terminate prior to that. As seen in (22) it contains the term $\ln(\det(\boldsymbol{R}))$. The correlation matrix $\boldsymbol{R}$ changes as a function of θ and $\det(\boldsymbol{R}) \to 0$ as $\boldsymbol{\theta} \to \infty$ and consequently $\ln(\det(\boldsymbol{R})) \to \infty$. Due to floating-point precision $\det(\boldsymbol{R})$ will be rounded off to zero when $\boldsymbol{\theta}$ reaches a certain size and as $\ln(0)$ is undefined, this will result in errors. Logically, an upper bound on $\boldsymbol{\theta}$ could be included in the optimization problem, but the point at which $\det(\boldsymbol{R}) = 0$ cannot be analytically ascertained. This problem can be relatively well handled by evolution strategies by resampling any infeasible point, but it can cause difficulties for gradient-based solvers.

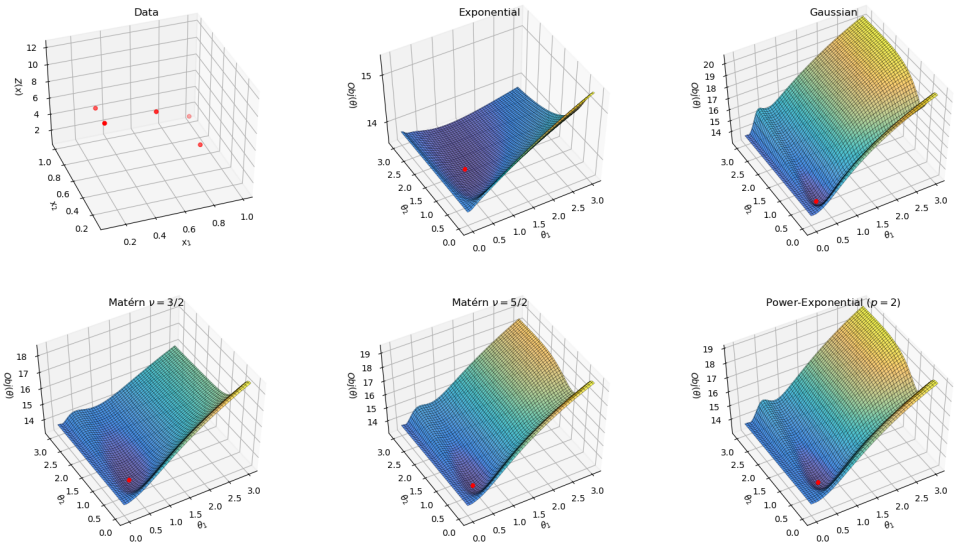


Figure 17: Illustration of the objective function and the optimum for the five possible correlation functions for a 2D ordinary kriging model. Top left shows the data the model is being fitted to.

Fig. 17 shows that already in the two-dimensional case the objective function of $\boldsymbol{\theta}$ can be a multi-modal problem. In the figure, the global minimum is shown, but there is an additional stationary area (valley) parallel to θ_2 where $\theta_1 \approx 0$.

Cross-Validation

7.3.4. Simple Kriging

In simple kriging, the trend, $\mu(\boldsymbol{x}, \boldsymbol{\beta})$ is constant and known, which is equivalent to $f_0(\boldsymbol{x}) = 1$ with $L = 0$ and $\beta_0 = \mu$ where μ is a known quantity, typically $\mu = \mathbb{E}[\boldsymbol{Z}]$.

Finding the optimal solution yielding the minimum prediction variance is straight foward for simple kriging. This is an unconstrained quadratic programming problem

$$\min_{\boldsymbol{\omega}} \quad \sigma^2 + \boldsymbol{\omega}^T \boldsymbol{C} \boldsymbol{\omega} - 2\boldsymbol{\omega}^T \boldsymbol{c}_0$$

which after differentiation w.r.t. $\boldsymbol{\omega}$ becomes

$$\boldsymbol{C}\boldsymbol{\omega}_{SK} = \boldsymbol{c}_0 \Leftrightarrow \boldsymbol{\omega}_{SK} = \boldsymbol{C}^{-1}\boldsymbol{c}_0$$

with the various entries being

$$\begin{bmatrix} C(\boldsymbol{x}_1 - \boldsymbol{x}_1) & \dots & C(\boldsymbol{x}_1 - \boldsymbol{x}_i) & \dots & C(\boldsymbol{x}_1 - \boldsymbol{x}_n) \\ \vdots & \dots & \vdots & \dots & \vdots \\ C(\boldsymbol{x}_i - \boldsymbol{x}_1) & \dots & C(\boldsymbol{x}_i - \boldsymbol{x}_i) & \dots & C(\boldsymbol{x}_i - \boldsymbol{x}_n) \\ \vdots & \dots & \vdots & \dots & \vdots \\ C(\boldsymbol{x}_n - \boldsymbol{x}_1) & \dots & C(\boldsymbol{x}_n - \boldsymbol{x}_i) & \dots & C(\boldsymbol{x}_n - \boldsymbol{x}_n) \end{bmatrix} \begin{bmatrix} \omega_1^{SK} \\ \vdots \\ \omega_i^{SK} \\ \vdots \\ \omega_n^{SK} \end{bmatrix} = \begin{bmatrix} C(\boldsymbol{x}_1 - \boldsymbol{x}_0) \\ \vdots \\ C(\boldsymbol{x}_i - \boldsymbol{x}_0) \\ \vdots \\ C(\boldsymbol{x}_n - \boldsymbol{x}_0) \end{bmatrix}$$

Notice here that $\boldsymbol{C}$ only requires inversion once.

The predictor for simple kriging is

$$Z^*_{\boldsymbol{\omega}_{SK}}(\boldsymbol{x}_0) = \mu + \sum_{i=1}^{n} \omega_i^{SK}(Z(\boldsymbol{x}_i) - \mu) = \mu + \boldsymbol{\omega}_{SK}^T(\boldsymbol{Z} - \mu\mathbb{1})$$

and the variance of the prediction error

$$\sigma^2_{SK} = \mathrm{Var}[Z^*_{\boldsymbol{\omega}_{SK}}(\boldsymbol{x}_0) - Z(\boldsymbol{x}_0)] = \sigma^2 - \sum_{i=1}^{n} \omega_i^{SK} C(\boldsymbol{x}_i - \boldsymbol{x}_0) = \sigma^2 - \boldsymbol{\omega}_{SK}^T \boldsymbol{c}_0$$

7.3.5. Ordinary Kriging

Ordinary kriging - similar to simple kriging - asumme a constant trend, but it is in this case unknown. So again $f_0(\boldsymbol{x}) = 1$ with $L = 0$, but β_0 has to be estimated.

The formulation of the minimum prediction variance is slightly different for ordinary kriging, namely due to the introduction of a constraint

$$\begin{aligned} \min_{\boldsymbol{\omega}} \quad & \boldsymbol{\omega}^T \boldsymbol{C} \boldsymbol{\omega} - 2\boldsymbol{\omega}^T \boldsymbol{c}_0 \\ \text{s.t.} \quad & \boldsymbol{\omega}^T \mathbb{1} = 1 \end{aligned}$$

this is an equality constrained problem and can be solved using the method of Lagrangian multipliers

$$\min_{\boldsymbol{\omega}} \quad \boldsymbol{\omega}^T \boldsymbol{C} \boldsymbol{\omega} - \mathbf{2}\boldsymbol{\omega}^{\boldsymbol{T}} \boldsymbol{c}_{\mathbf{0}} - 2\lambda(\boldsymbol{w}^T \mathbb{1} - 1)$$

which can be written as a KKT system for a quadratic programming problem

$$\tilde{\boldsymbol{C}} \boldsymbol{\omega}_{OK} = \tilde{\boldsymbol{c}}_0 \Leftrightarrow \boldsymbol{\omega}_{OK} = \tilde{\boldsymbol{C}}^{-1} \tilde{\boldsymbol{c}}_0$$

with the various entries being

$$\begin{bmatrix} C(\boldsymbol{x}_1 - \boldsymbol{x}_1) & \dots & C(\boldsymbol{x}_1 - \boldsymbol{x}_i) & \dots & C(\boldsymbol{x}_1 - \boldsymbol{x}_n) & 1 \\ \vdots & \dots & \vdots & \dots & \vdots & \\ C(\boldsymbol{x}_i - \boldsymbol{x}_1) & \dots & C(\boldsymbol{x}_i - \boldsymbol{x}_i) & \dots & C(\boldsymbol{x}_i - \boldsymbol{x}_n) & 1 \\ \vdots & \dots & \vdots & \dots & \vdots & \\ C(\boldsymbol{x}_n - \boldsymbol{x}_1) & \dots & C(\boldsymbol{x}_n - \boldsymbol{x}_i) & \dots & C(\boldsymbol{x}_n - \boldsymbol{x}_n) & 1 \\ 1 & \dots & 1 & \dots & 1 & 0 \end{bmatrix} \begin{bmatrix} \omega_1^{OK} \\ \vdots \\ \omega_i^{OK} \\ \vdots \\ \omega_n^{OK} \\ -\lambda_{OK} \end{bmatrix} = \begin{bmatrix} C(\boldsymbol{x}_1 - \boldsymbol{x}_0) \\ \vdots \\ C(\boldsymbol{x}_i - \boldsymbol{x}_0) \\ \vdots \\ C(\boldsymbol{x}_n - \boldsymbol{x}_0) \\ 1 \end{bmatrix}$$

where λ_{OK} is a Lagrangian multiplier.

The predictor for ordinary kriging is

$$Z^*_{\boldsymbol{\omega}_{OK}}(\boldsymbol{x}_0) = \sum_{i=1}^{n} \omega_i^{OK} Z(\boldsymbol{x}_i) = \boldsymbol{\omega}_{OK}^T \boldsymbol{Z}$$

and the variance of the prediction error

$$\sigma^2_{OK} = \mathrm{Var}[Z^*_{\boldsymbol{\omega}_{OK}}(\boldsymbol{x}_0) - Z(\boldsymbol{x}_0)] = \sigma^2 - \sum_{i=1}^{n} \omega_i^{OK} C(\boldsymbol{x}_i - \boldsymbol{x}_0) = \sigma^2 - \boldsymbol{\omega}_{OK}^T \boldsymbol{c}_0 + \lambda_{OK}$$

7.3.6. Universal Kriging

Universal kriging - unlike the two others - does not assume a constant trend. In this case the trend, $\mu(\boldsymbol{x};\boldsymbol{\theta})$ may be a linear combination of an arbitrary number of $f_l(\boldsymbol{x})$ for $l = 1, ..., L$ and with $f_0(\boldsymbol{x}) = 1$.

The formulation of the minimum prediction for universal kriging, is also a constrained qudratic programming problem

$$\begin{aligned} \min_{\boldsymbol{\omega}} \quad & \boldsymbol{\omega}^T \boldsymbol{C}\boldsymbol{\omega} - 2\boldsymbol{\omega}^T \boldsymbol{c}_0 \\ \text{s.t.} \quad & \boldsymbol{F}^T\boldsymbol{\omega} = \boldsymbol{f}_0 \end{aligned}$$

this is an equality constrained problem and can be solved using the method of Lagrangian multipliers

$$\min_{\boldsymbol{\omega}} \quad \boldsymbol{\omega}^T \boldsymbol{C}\boldsymbol{\omega} - \mathbf{2}\boldsymbol{\omega}^{\boldsymbol{T}} \boldsymbol{c_0} - 2\boldsymbol{\lambda}(\boldsymbol{F}\boldsymbol{w} - \boldsymbol{f}_0)$$

which can be written as a KKT system for a quadratic programming problem

$$\begin{bmatrix} \boldsymbol{C} & \boldsymbol{F} \\ \boldsymbol{F}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_{UK} \\ -\boldsymbol{\lambda}_{UK} \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_0 \\ \boldsymbol{f}_0 \end{bmatrix}$$

with the various entries being

$$\begin{bmatrix} C(\boldsymbol{x}_1 - \boldsymbol{x}_1) & \dots & C(\boldsymbol{x}_1 - \boldsymbol{x}_n) & 1 & f_1(\boldsymbol{x}_1) & \dots & f_L(\boldsymbol{x}_1) \\ C(\boldsymbol{x}_2 - \boldsymbol{x}_1) & \dots & C(\boldsymbol{x}_2 - \boldsymbol{x}_n) & 1 & f_1(\boldsymbol{x}_2) & \dots & f_L(\boldsymbol{x}_2) \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ C(\boldsymbol{x}_n - \boldsymbol{x}_1) & \dots & C(\boldsymbol{x}_n - \boldsymbol{x}_n) & 1 & f_1(\boldsymbol{x}_n) & \dots & f_L(\boldsymbol{x}_n) \\ 1 & \dots & 1 & 0 & 0 & \dots & 0 \\ f_1(\boldsymbol{x}_1) & \dots & f_1(\boldsymbol{x}_n) & 0 & 0 & \dots & 0 \\ \vdots & \dots & \vdots & \dots & \vdots & \dots & \vdots \\ f_n(\boldsymbol{x}_1) & \dots & f_n(\boldsymbol{x}_n) & 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} \omega_1^{UK} \\ \omega_2^{UK} \\ \vdots \\ \omega_n^{UK} \\ -\lambda_0^{UK} \\ -\lambda_1^{UK} \\ \vdots \\ -\lambda_n^{UK} \end{bmatrix} = \begin{bmatrix} C(\boldsymbol{x}_1 - \boldsymbol{x}_0) \\ C(\boldsymbol{x}_2 - \boldsymbol{x}_0) \\ \vdots \\ C(\boldsymbol{x}_n - \boldsymbol{x}_0) \\ 1 \\ f_1(\boldsymbol{x}_0) \\ \vdots \\ f_n(\boldsymbol{x}_0) \end{bmatrix}$$

where $\boldsymbol{\lambda}_{UK}$ are Lagrangian multipliers.

The predictor for universal kriging is

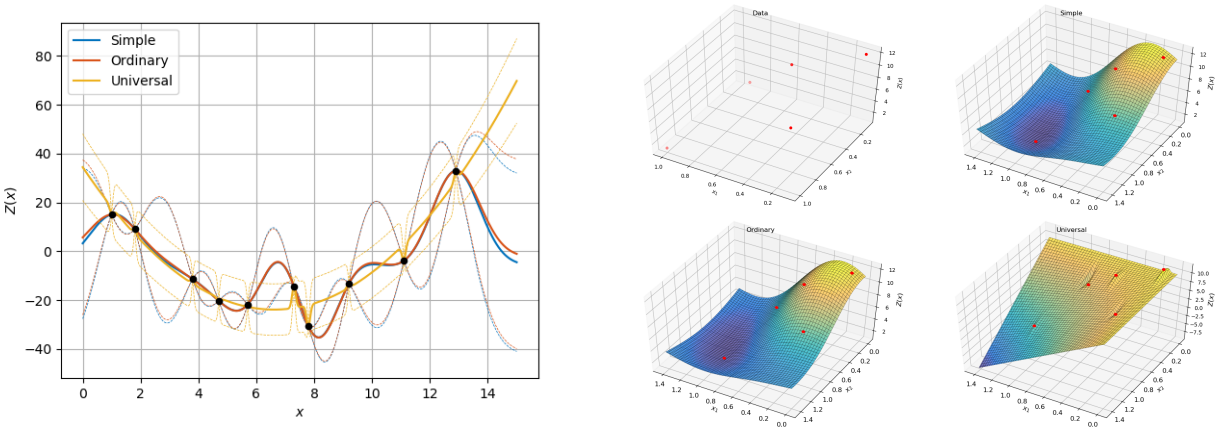
$$Z^*_{\boldsymbol{\omega}_{UK}}(\boldsymbol{x}_0) = \sum_{i=1}^{n} \omega_i^{UK} Z(\boldsymbol{x}_i) = \boldsymbol{\omega}_{UK}^T \boldsymbol{Z}$$

and the variance of the prediction error

$$\sigma^2_{UK} = \mathrm{Var}[Z^*_{\boldsymbol{\omega}_{UK}}(\boldsymbol{x}_0) - Z(\boldsymbol{x}_0)] = \sigma^2 - \sum_{i=1}^{n} \omega_i^{UK} C(\boldsymbol{x}_i - \boldsymbol{x}_0) + \sum_{l=0}^{L} f_l(\boldsymbol{x}_0)\lambda_l^{UK} = \sigma^2 - \boldsymbol{\omega}_{UK}^T \boldsymbol{c}_0 + \boldsymbol{f}_0^T \boldsymbol{\lambda}_{UK}$$

7.3.7. Kriging Type Comparison

Simple and Ordinary kriging are very similar. They are both mean-reverting when extrapolating and performs smooth interpolations (dependent on the covariogram). Universal kriging behaves differently due to the underlying assumption of a position dependent trend. It extrapolates according to that trend and also has a more fluctuating interpolation. This fluctuation is due to the additional impact of the trend often leads to lower estimates of $\boldsymbol{\theta}$ resulting in rapidly declining covariance functions. As seen from Fig. 18a the error estimate simple and ordinary kriging behave in a similar fashion, with ordinary kriging having a slightly higher uncertainty. This is analogue to the results of bias-correction in maximum likelihood and restricted maximum likelihood in the estimate of the variance parameter of a normal distribution. It is related to the fact that the



(a) 1D (b) 2D

Figure 18: Illustration of Simple, Ordinary and Universal Kriging in 1D (left) and 2D (right). The trend of the universal kriging is $f_0(\boldsymbol{x}) = 1$ and $f_1(\boldsymbol{x}) = x_1 x_2$.

trend is assumed known for simple kriging, but unknown for ordinary kriging thus adding uncertainty due to the uncertainty on the estimate of β_0. The uncertainty for universal kriging is similar to the estimate also heavily influenced by the underlying trend assumption.

Fig. 18b illustrates a potential downside of universal kriging for use in proxy-modelling. Because kriging is an exact interpolator it will precisely honor the simulated points creating multiple minor local minimums (or maximums) which are of no practical value in the exploration process. This can cause problems for the optimization algorithms used to explore the solution space.

In conclusion universal kriging relies too heavily on the assumption of the underlying trend model or it to be useful in proxy-modelling. The assumption of a known trend is in practice not feasible which rules out simple kriging. Consequently, the preferred assumption should be ordinary kriging.

7.4. Radial Basis Function Interpolation

Interpolation by use of Radial Basis Functions (RBF) is a linear estimator

$$Z^*_{\boldsymbol{\omega}}(\boldsymbol{x}_0) = \sum_{i=1}^{n} \omega_i \varphi(||\boldsymbol{x}_0 - \boldsymbol{x}_i||) \tag{23}$$

where $\boldsymbol{x}_0$ is the interpolated point, $\boldsymbol{x}_i$ is the position of an observed point, with a corresponding observation $Z(\boldsymbol{x}_i)$, $\varphi : [0, \infty] \to \mathbb{R}$ is a radial basis function (called *kernel*) which takes the input $||\cdot|| : \boldsymbol{v} \to [0, \infty]$ and $\omega_i \in \mathbb{R}$ is a weight. Notice here that the input to $\varphi(\cdot)$ is the norm of the distance vector. This means that any variable $\boldsymbol{x}$ has to be mapped to a uniform vector space such as $\boldsymbol{x} \in [0, 1]^d$ to ensure equal length scales in all dimensions.

7.4.1. Radial Basis Function

The suite of available kernel functions for RBF interpolation is

$$\begin{aligned}
\text{Linear:} \quad & \varphi(r;\kappa) = \kappa r \\
\text{Cubic:} \quad & \varphi(r;\kappa) = (\kappa r)^3 \\
\text{Thin-Plate:} \quad & \varphi(r;\kappa) = (\kappa r)^2 \ln(\kappa r) \\
\text{Multi Quadratic:} \quad & \varphi(r;\kappa) = \sqrt{(\kappa r)^2 + 1} \\
\text{Inverse Quadratic:} \quad & \varphi(r;\kappa) = \frac{1}{1 + (\kappa r)^2} \\
\text{Inverse Multi Quadratic:} \quad & \varphi(r;\kappa) = \frac{1}{\sqrt{1 + (\kappa r)^2}} \\
\text{Gaussian:} \quad & \varphi(r;\kappa) = e^{-(\kappa r)^2}
\end{aligned}$$

where $r = ||\boldsymbol{x}_i - \boldsymbol{x}_j||$ and $\kappa > 0$ is a shape parameter.

7.4.2. Hyper-Parameter Estimation

Hyper-parameter estimation of the scale parameter, κ, is typically done using a cross-validation estimator. In this case a K-fold cross-validation technique is used and the estimator of κ is [12]

$$\begin{aligned}
\min_{\kappa} \quad & \sum_{i=1}^{K} \left(Z(\boldsymbol{x}_i) - Z^*_{\boldsymbol{\omega},(-i)}(\boldsymbol{x}_i) \right)^2 \\
\text{s.t.} \quad & \kappa > 1
\end{aligned}$$

where $Z(\boldsymbol{x}_i)$ is the known observation at $\boldsymbol{x}_i$ and $Z^*_{\boldsymbol{\omega},(-i)}(\boldsymbol{x}_i)$ is the prediction at $\boldsymbol{x}_i$ using weights, $\boldsymbol{\omega}$, obtained from the sub-set of observations without the i'th observation. Notice that when $K = n$ (Leave-one-out cross-validation) then i is an index. When $K < n$ then i is a set of indices. Cross-validation will be explained in more detail later where and will also be used for testing the predictability of models.

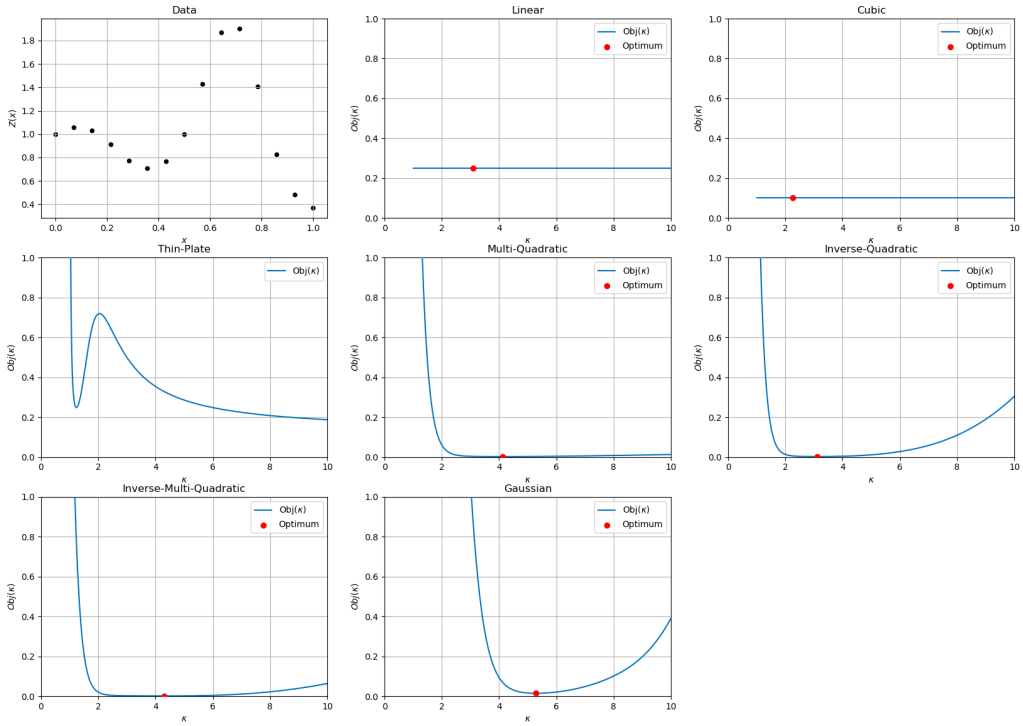


Figure 19: Illustration of the objective function and the optimum for the seven possible RBFs for a 1D scenario. Top left shows the data the model is being fitted to.

Fig. 19 shows a few interesting things. First of all it is evident that the linear and cubic RBFs are independent of the shape parameter, κ. Further, the thin-plate RBF has a very different curvature then the remaining, including a local optimum around $\kappa = 1.3$ followed by a long tail, of which the optimum is actually found at $\kappa = 45681$. The remainder of the RBFs have a more convex behaviour w.r.t. the objective function with a single global optimum.

7.4.3. Interpolation

The weights, ω_i, used in the interpolation are determined by solving the following system of equations

$$\boldsymbol{\Phi\omega} = \boldsymbol{Z} \Leftrightarrow \boldsymbol{\omega} = \boldsymbol{\Phi}^{-1}\boldsymbol{Z}$$

with the various entries being

$$\begin{bmatrix} \varphi(||\boldsymbol{x}_1 - \boldsymbol{x}_1||) & \dots & \varphi(||\boldsymbol{x}_1 - \boldsymbol{x}_i||) & \dots & \varphi(||\boldsymbol{x}_1 - \boldsymbol{x}_n||) \\ \vdots & \dots & \vdots & \dots & \vdots \\ \varphi(||\boldsymbol{x}_i - \boldsymbol{x}_1||) & \dots & \varphi(||\boldsymbol{x}_i - \boldsymbol{x}_i||) & \dots & \varphi(||\boldsymbol{x}_i - \boldsymbol{x}_n||) \\ \vdots & \dots & \vdots & \dots & \vdots \\ \varphi(||\boldsymbol{x}_n - \boldsymbol{x}_1||) & \dots & \varphi(||\boldsymbol{x}_n - \boldsymbol{x}_i||) & \dots & \varphi(||\boldsymbol{x}_n - \boldsymbol{x}_n||) \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_i \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} Z(\boldsymbol{x}_1) \\ \vdots \\ Z(\boldsymbol{x}_i) \\ \vdots \\ Z(\boldsymbol{x}_n) \end{bmatrix}$$

The weights are uniquely defined by the observed points and consequently independent of the point at which prediction occurs. Once the weights are found prediction can be made using (23). Often multiple predictions are made simultaneously which can be easily predicted by constructing a similar system, but with distances

relative to $\boldsymbol{x}_0$ (of size $p \times m$) where p is the number of points to predict.

$$\boldsymbol{\Phi}_0 \boldsymbol{\omega} = \boldsymbol{Z}^*_{\boldsymbol{\omega}}$$

where the various entries are

$$\begin{bmatrix} \varphi(||\boldsymbol{x}_0^{(0)} - \boldsymbol{x}_1||) & \dots & \varphi(||\boldsymbol{x}_0^{(0)} - \boldsymbol{x}_i||) & \dots & \varphi(||\boldsymbol{x}_0^{(0)} - \boldsymbol{x}_n||) \\ \vdots & \dots & \vdots & \dots & \vdots \\ \varphi(||\boldsymbol{x}_0^{(j)} - \boldsymbol{x}_1||) & \dots & \varphi(||\boldsymbol{x}_0^{(j)} - \boldsymbol{x}_i||) & \dots & \varphi(||\boldsymbol{x}_0^{(j)} - \boldsymbol{x}_n||) \\ \vdots & \dots & \vdots & \dots & \vdots \\ \varphi(||\boldsymbol{x}_0^{(p)} - \boldsymbol{x}_1||) & \dots & \varphi(||\boldsymbol{x}_0^{(p)} - \boldsymbol{x}_i||) & \dots & \varphi(||\boldsymbol{x}_0^{(p)} - \boldsymbol{x}_n||) \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_i \\ \vdots \\ \omega_n \end{bmatrix} = \begin{bmatrix} Z^*_{\boldsymbol{\omega}}(\boldsymbol{x}_0^{(0)}) \\ \vdots \\ Z^*_{\boldsymbol{\omega}}(\boldsymbol{x}_0^{(j)}) \\ \vdots \\ Z^*_{\boldsymbol{\omega}}(\boldsymbol{x}_0^{(p)}) \end{bmatrix}$$

7.5. Polynomial Chaos Expansion

7.6. Neural-Network Models

7.7. Cross-Validation

8. Optimization Algorithms

There are several optimization algorithms implemented in `modpy` to solve the following problems

- Linear Programming (LP)
- Quadratic Programming (QP)
- Non-Linear Programming (NLP)
- Multi-Modal Optimization (MMO)
- Multi-Objective Optimization (MOO)

For the most common issue, namely NLP, there are several available algorithms such as

- Sequential Quadratic Programming Method (SQP)
- Interior-Point Method (IP)
- Covariance Matrix Adaptation Evolution Strategies (CMA-ES)
- Hybrid Method

where SQP and IP are gradient-based methods and CMA-ES is a stochastic method. Hybrid Method is a combination of the two.

8.1. Linear Programming

8.2. Quadratic Programming

Given an optimization problem of the form

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} + \boldsymbol{g}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{C}\boldsymbol{x} \geq \boldsymbol{d} \end{aligned} \tag{24}$$

it may be solved using a quadratic programming algorithm. Quadratic programming problems can be split into five different cases. Let $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ then

- $\boldsymbol{H}$ is positive definite, denoted by $\boldsymbol{H} \succ 0$, if $\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} \geq \boldsymbol{0}$ for all $\boldsymbol{x} \in \mathbb{R}^n$
- $\boldsymbol{H}$ is positive semi-definite, denoted by $\boldsymbol{H} \succeq 0$, if $\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} > \boldsymbol{0}$ for all $\boldsymbol{0} \neq \boldsymbol{x} \in \mathbb{R}^n$
- $\boldsymbol{H}$ is negative semi-definite, denoted by $\boldsymbol{H} \preceq 0$, if $\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} < \boldsymbol{0}$ for all $\boldsymbol{0} \neq \boldsymbol{x} \in \mathbb{R}^n$
- $\boldsymbol{H}$ is negative definite, denoted by $\boldsymbol{H} \prec 0$, if $\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} \leq \boldsymbol{0}$ for all $\boldsymbol{x} \in \mathbb{R}^n$
- $\boldsymbol{H}$ is indefinite if there exists an $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^n$ such that $\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} > 0$ and $\boldsymbol{y}^T\boldsymbol{H}\boldsymbol{y} < \boldsymbol{0}$.

If $\boldsymbol{H}$ is symmetric and positive definite then (24) is a strictly convex problem with a single and global optimum. For the positive- semi-definite case there will be multiple global optimums of equal value, and consequently the solution is non-unique. For the negative definite and semi-definite case the problem is strictly concave and concave respectively. In such situations the solution is no longer found in the interior of the solution space, but rather at the boundaries. For the indefinite case there exists saddle points as well as local and global optimums. The quadratic programming solvers implemented in `modpy` assumes that $\boldsymbol{H} \succ \boldsymbol{0}$ which guarantess a single and global optimum.

In order to solve a constrained optimization problem the Karush-Kuhn-Tucker (KKT) conditions have to be satisfied

$$\begin{aligned}
&\nabla_x \mathcal{L}(x, \lambda) = \nabla f(x) - \sum_{i=1}^{m} \lambda_i \nabla c_i(x) = 0 \\
&c_i(x) = 0, \quad i \in \mathcal{E} \\
&c_i(x) \geq 0, \quad i \in \mathcal{I} \\
&\lambda_i \geq 0, \quad i \in \mathcal{I} \\
&\lambda_i = 0 \vee c_i(x) = 0, \quad i \in \mathcal{I}
\end{aligned}$$

where the last condition is called the complimentary condition.

For the unconstrained and the purely equality constrained problem there is a closed-form solution to the problem. If inequality constraints are included it requires an iterative solver.

8.2.1. Unconstrained

In the unconstrained case the solution to the quadratic programming problem is simply

$$\boldsymbol{H}\boldsymbol{x} = \boldsymbol{g} \Leftrightarrow \boldsymbol{x} = \boldsymbol{H}^{-1}\boldsymbol{g}$$

which is equivalent to differentiating the objective function, setting it equal to zero and solving it. Given that $\boldsymbol{H}$ is symmetric and positive definite, this can be solved efficiently using Cholesky Factorization.

8.2.2. Equality Constrained

In the equality constrained case, the Lagrangian equation is solved instead of directly solving the quadratic equation.

$$\mathcal{L}(x, \lambda) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{g}^T\boldsymbol{x} - \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$$

which is similarly to the unconstrained problem differentiated w.r.t. $\boldsymbol{x}$, set equal to zero and solved. Let

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{H} & -\boldsymbol{A}^T \\ -\boldsymbol{A} & 0 \end{bmatrix}, \quad \boldsymbol{b}_{\mathcal{L}} = \begin{bmatrix} -\boldsymbol{g} \\ -\boldsymbol{b} \end{bmatrix}, \quad \boldsymbol{x}_{\mathcal{L}} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix}$$

then the solution to the problem is

$$\boldsymbol{K}\boldsymbol{x}_{\mathcal{L}} = \boldsymbol{b}_{\mathcal{L}} \Leftrightarrow \boldsymbol{x}_{\mathcal{L}} = \boldsymbol{K}^{-1}\boldsymbol{b}_{\mathcal{L}}$$

In this case K is non-singular, symmetric and indefinite. Due to this it is preferable to use an LDL Factorization which utilizes exactly those properties of the matrix.

8.2.3. Inequality Constrained

There are two families of inequality constrained solvers, namely Active Set methods and Interior Point methods. The algorithm implemented in `modpy` is a Primal-Dual Predictor Corrector Interior Point Algorithm based on an extension to Mehrotra's algorithm for linear programming. Similar to the equality constrained case it is the Lagrangian equation that is solved

$$\mathcal{L}(x, \lambda) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{g}^T\boldsymbol{x} - \boldsymbol{\lambda}_{\mathcal{E}}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) - \boldsymbol{\lambda}_{\mathcal{I}}^T(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{d} + \boldsymbol{s})$$

The algorithm iteratively improves the candidate solution until certain termination criterias are met. Define the residuals

$$\begin{aligned}\boldsymbol{r}_{\mathcal{L}} &= \boldsymbol{H}\boldsymbol{x} + \boldsymbol{g} - \boldsymbol{A}^T\boldsymbol{\lambda}_{\mathcal{E}} - \boldsymbol{C}^T\boldsymbol{\lambda}_{\mathcal{I}} \\ \boldsymbol{r}_A &= \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \\ \boldsymbol{r}_C &= \boldsymbol{s} + \boldsymbol{d} - \boldsymbol{C}\boldsymbol{x}\end{aligned}$$

where $\boldsymbol{\lambda}_{\mathcal{E}}$ and $\boldsymbol{\lambda}_{\mathcal{I}}$ are Lagrangian multipliers of the equality and inequality constraint respectively and $\boldsymbol{s}$ are slack variables.

First an affine scaling step is calculated using an augmented system which eliminates the inequality constraints, reducing the computational requirement when m_i is large. The KKT system is assembled as follows

$$\boldsymbol{K} = \begin{bmatrix}\bar{\boldsymbol{H}} & -\boldsymbol{A}^T \\ -\boldsymbol{A} & 0\end{bmatrix}, \quad \boldsymbol{b}_{\mathcal{L}} = \begin{bmatrix}-\bar{\boldsymbol{r}}_{\mathcal{L}} \\ -\boldsymbol{r}_A\end{bmatrix}, \quad \Delta\boldsymbol{x}_{\mathcal{L}}^{\text{aff}} = \begin{bmatrix}\Delta\boldsymbol{x}^{\text{aff}} \\ \Delta\boldsymbol{\lambda}_{\mathcal{E}}^{\text{aff}}\end{bmatrix} \tag{25}$$

define the two diagonal matrices of inequality Lagrangian multipliers and slack variables

$$\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_{1,\mathcal{I}}, \lambda_{2,\mathcal{I}}, \ldots, \lambda_{m_i,\mathcal{I}}), \quad \boldsymbol{S} = \operatorname{diag}(s_1, s_2, \ldots, s_{m_i})$$

then $\bar{\boldsymbol{H}}$ and $\bar{\boldsymbol{r}}_{\mathcal{L}}$ may be defined as

$$\begin{aligned}\bar{\boldsymbol{H}} &= \boldsymbol{H} + \boldsymbol{C}^T\boldsymbol{\Lambda}\boldsymbol{S}^{-1}\boldsymbol{C} \\ \bar{\boldsymbol{r}}_{\mathcal{L}} &= \boldsymbol{r}_{\mathcal{L}} - \boldsymbol{C}^T\Lambda\boldsymbol{S}^{-1}(\boldsymbol{r}_C - \boldsymbol{s})\end{aligned}$$

the KKT system has the solution

$$\boldsymbol{K}\Delta\boldsymbol{x}_{\mathcal{L}}^{\text{aff}} = \boldsymbol{b}_{\mathcal{L}} \Leftrightarrow \Delta\boldsymbol{x}_{\mathcal{L}}^{\text{aff}} = \boldsymbol{K}^{-1}\boldsymbol{b}_{\mathcal{L}} \tag{26}$$

which can be solved using a modified Cholesky factorization. Using the solution to the KKT system, the Lagrangian multipliers of the inequality constrained system and the slack variables can be updated

$$\begin{aligned}\Delta\boldsymbol{\lambda}_{\mathcal{I}}^{\text{aff}} &= -\boldsymbol{\Lambda}\boldsymbol{S}^{-1}\left(\boldsymbol{C}\Delta\boldsymbol{x}^{\text{aff}} + \boldsymbol{r}_C - \boldsymbol{s}\right) \\ \Delta\boldsymbol{s}^{\text{aff}} &= -\boldsymbol{s} - \boldsymbol{S}\boldsymbol{\Lambda}^{-1}\boldsymbol{\lambda}_{\mathcal{I}}^{\text{aff}}\end{aligned}$$

Then an affine step-length selection is made which satisfies

$$\alpha^{\text{aff}} = \max\left\{\alpha \in (0,\, 1] \,\middle|\, \boldsymbol{\lambda}_{\mathcal{I}} + \alpha\Delta\boldsymbol{\lambda}_{\mathcal{I}}^{\text{aff}} \geq 0 \wedge \boldsymbol{s} + \alpha\Delta\boldsymbol{s}^{\text{aff}} \geq 0\right\}$$

this leads to an update of the affine penalty parameter

$$\mu^{\text{aff}} = \frac{1}{m_i} \cdot \left(\boldsymbol{x} + \alpha^{\text{aff}}\Delta\boldsymbol{x}^{\text{aff}}\right)^T\left(\boldsymbol{s} + \alpha^{\text{aff}}\Delta\boldsymbol{s}^{\text{aff}}\right)$$

and the heuristic for the centering parameter

$$\sigma = \left(\frac{\mu^{\text{aff}}}{\mu}\right)^3$$

Once all the affine step parameters and the centering parameter are estimated, the components of the right-hand side of (28) may then be updated

$$\bar{\boldsymbol{r}}_{\mathcal{L}} = \boldsymbol{r}_{\mathcal{L}} - \boldsymbol{C}^T \Lambda \boldsymbol{S}^{-1} \left(\boldsymbol{r}_C - \boldsymbol{\Lambda}^{-1} \left(\boldsymbol{\Lambda}\boldsymbol{S}\boldsymbol{e} + \Delta\boldsymbol{\Lambda}^{\text{aff}}\Delta\boldsymbol{S}^{\text{aff}}\boldsymbol{e} - \sigma\mu\boldsymbol{e}\right)\right) \tag{27}$$

where $\boldsymbol{e}$ is a unit vector and the system is solved again

$$\boldsymbol{K}\Delta\boldsymbol{x}_{\mathcal{L}} = \boldsymbol{b}_{\mathcal{L}} \Leftrightarrow \Delta\boldsymbol{x}_{\mathcal{L}} = \boldsymbol{K}^{-1}\boldsymbol{b}_{\mathcal{L}} \tag{28}$$

where $\boldsymbol{K}$ is the same as in (25), $\boldsymbol{b}_{\mathcal{L}}$ is updated with (27) and $\Delta\boldsymbol{x}_{\mathcal{L}}$ has the same form as (25), but is no longer the affine step, but rather the corrected step. Due to using the augmented formulation, the step for the inequality Lagrangian multiplier has to be backsubstituted

$$\Delta\boldsymbol{\lambda}_{\mathcal{I}} = -\boldsymbol{\Lambda}\boldsymbol{S}\left(\boldsymbol{C}\Delta\boldsymbol{x} - \left(\boldsymbol{r}_C - \boldsymbol{\Lambda}^{-1}\left(\boldsymbol{\Lambda}\boldsymbol{S}\boldsymbol{e} + \Delta\boldsymbol{\Lambda}^{\text{aff}}\Delta\boldsymbol{S}^{\text{aff}}\boldsymbol{e} - \sigma\mu\boldsymbol{e}\right)\right)\right)$$

and similarly for the slack variable

$$\Delta\boldsymbol{s} = -\boldsymbol{\Lambda}^{-1}\left(\boldsymbol{\Lambda}\boldsymbol{S}\boldsymbol{e} + \Delta\boldsymbol{\Lambda}^{\text{aff}}\Delta\boldsymbol{S}^{\text{aff}}\boldsymbol{e} - \sigma\mu\boldsymbol{e}\right) - \boldsymbol{\Lambda}\boldsymbol{S}\Delta\boldsymbol{\lambda}_{\mathcal{I}}$$

The optimal primal and dual step-lengths are chosen

$$\alpha_{\text{prim}} = \min\left(1, \left\{-\frac{\boldsymbol{\lambda}_{\mathcal{I}}}{\Delta\boldsymbol{\lambda}_{\mathcal{I}}^{\text{aff}}} \;\middle|\; \boldsymbol{\lambda}_{\mathcal{I}}^{\text{aff}} < 0\right\}\right)$$
$$\alpha_{\text{dual}} = \min\left(1, \left\{-\frac{\boldsymbol{s}}{\Delta\boldsymbol{s}^{\text{aff}}} \;\middle|\; \boldsymbol{s}^{\text{aff}} < 0\right\}\right)$$

unlike in the linear programming case it can cause the algorithm to diverge for certain combinations of α_{prim} and α_{dual}. There are methods for overcoming this issue while retaining seperate step-lengths for the primal and the dual problem, but `modpy` uses the simplified solution of

$$\alpha = \min\left(\alpha_{\text{prim}}^{\text{aff}}, \alpha_{\text{dual}}^{\text{aff}}\right)$$

Based on the solved steps and the step-length the solution vectors can be updated

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha\Delta\boldsymbol{x}$$
$$\boldsymbol{\lambda}_{\mathcal{E}}^{k+1} = \boldsymbol{\lambda}_{\mathcal{E}}^k + \alpha\Delta\boldsymbol{\lambda}_{\mathcal{E}}$$
$$\boldsymbol{\lambda}_{\mathcal{I}}^{k+1} = \boldsymbol{\lambda}_{\mathcal{I}}^k + \alpha\Delta\boldsymbol{\lambda}_{\mathcal{I}}$$
$$\boldsymbol{s}^{k+1} = \boldsymbol{s}^k + \alpha\Delta\boldsymbol{s}$$

where k denotes the iteration counter. Finally the residuals are updated with the new solution.

The variables are iteratively updated until he following criteria is satisfied

$$||\boldsymbol{r}_{\mathcal{L}}||_\infty \leq \epsilon_{\mathcal{L}} \cdot \max\left(1, ||\left[\boldsymbol{H}, \boldsymbol{g}^T, \boldsymbol{A}^T, \boldsymbol{C}^T\right]||_\infty\right)$$
$$||\boldsymbol{r}_A||_\infty \leq \epsilon_A \cdot \max\left(1, ||\left[\boldsymbol{A}, \boldsymbol{b}^T\right]||_\infty\right)$$
$$||\boldsymbol{r}_C||_\infty \leq \epsilon_C \cdot \max\left(1, ||\left[\boldsymbol{I}, \boldsymbol{C}, \boldsymbol{d}^T\right]||_\infty\right)$$
$$|\mu| \leq \epsilon_\mu \cdot 0.01 \cdot \mu_0$$

where $|||[\cdot]|||$ denotes the matrix-norm over the block-matrix encapsulaed in the brackets $[\cdot]$. ϵ_α $\alpha \in \{\mathcal{L}, A, C, \mu\}$ defines the tolerances for the objective function, equality constraints, inequality constraints and penalty parameter respectively. μ is a penalty parameter and μ_0 is the initial penalty parameter.

8.3. Nonlinear Programming

The general nonlinear programming problem (NLP) can be formulated as

$$\begin{aligned} \min_{x} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & c_i(\boldsymbol{x}) = 0, \quad i \in \mathcal{E} \\ & c_i(\boldsymbol{x}) \geq 0, \quad i \in \mathcal{I} \end{aligned} \tag{29}$$

There are two primary methods of solving NLPs, Sequential Quadratic Programming methods and Interior-Point methods.

8.3.1. Sequential Quadratic Programming

The logic of sequantial quadratic programming (SQP) methods is to linearize the nonlinear problem using a 2^{nd}-order Taylor expansion for the objective function and 1^{st}-order for the constraints which takes the form of a QP. The NLP is then solved iteratively for a direction, $\boldsymbol{p}$, by solving a QP at each iteration which is added to the solution vector, similar to a Newton method. The quadratic sub-problem has the form

$$\begin{aligned} \min_{p} \quad & f_k + \nabla f_k^T \boldsymbol{p} + \frac{1}{2} \boldsymbol{p}^T \nabla^2_{xx} \mathcal{L}_k \boldsymbol{p} \\ \text{s.t.} \quad & \nabla c_i(\boldsymbol{x}_k)^T \boldsymbol{p} + c_i(\boldsymbol{x}_k) = 0, \quad i \in \mathcal{E} \\ & \nabla c_i(\boldsymbol{x}_k)^T \boldsymbol{p} + c_i(\boldsymbol{x}_k) = 0, \quad i \in \mathcal{I} \end{aligned} \tag{30}$$

which can be shown to have the same form as (24) with

$$f_k = 0, \quad \nabla f_k^T = \boldsymbol{g}, \quad \nabla^2_{xx} \boldsymbol{\mathcal{L}}_k = \boldsymbol{H}, \quad \nabla c_{\mathcal{E}}(\boldsymbol{x}_k)^T = \boldsymbol{A}, \quad c_{\mathcal{E}}(\boldsymbol{x}_k) = -\boldsymbol{b}, \quad \nabla c_{\mathcal{I}}(\boldsymbol{x}_k) = \boldsymbol{C}, \quad c_{\mathcal{I}}(\boldsymbol{x}_k) = -\boldsymbol{d}$$

The SQP method implemented in `modpy` is similar to that of `SNOPT` [8] which first solves an LP to detect infeasible linear constraints. If the linear part of the constraints are feasible it proceeds to solve the non-linear problem (30) using (31). If the optimizer encounters an infeasible linearization of the non-linear constraints it changes to a so-called *elastic mode* and solves (32) instead. This formulation is an $\ell 1$ penalty problem used to ensure the quadratic sub-problem remains consistent after linearization of the constraints.

Linear Program

An LP is solved to check for infeasibility of the linear constraints of the problem.

$$\begin{aligned} \min_{x,v,w} \quad & \boldsymbol{e}^T(\boldsymbol{v} + \boldsymbol{w}) \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} - \boldsymbol{v} + \boldsymbol{w} \leq \boldsymbol{b} \\ & \boldsymbol{v}, \boldsymbol{w} \geq \boldsymbol{0} \end{aligned}$$

where $\boldsymbol{e}$ is a vector of ones. This LP is equivalent to minimizing the sum of the linear constraint violations. If $\boldsymbol{v} \neq \boldsymbol{0}$ or $\boldsymbol{w} \neq \boldsymbol{0}$ the linear constraints are infeasible and the algorithm terminates. The LP can be formulated in linear algebra formulation which can be passed to an LP algorithm

$$\boldsymbol{g} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{e} \\ \boldsymbol{e} \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_L & -\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{b} = \boldsymbol{b}_L, \quad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}_L & \boldsymbol{0} & -\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{d} = \boldsymbol{d}_L$$

Standard SQP

As mentioned the standard SQP uses the formulation (30) and consequently the linear algebra formulation of the problem is given as follows

$$\boldsymbol{H} = \boldsymbol{B}_k, \quad \boldsymbol{g} = \nabla f(\boldsymbol{x}_k)^T, \quad \boldsymbol{A} = \nabla c_{\mathcal{E}}(\boldsymbol{x}_k), \quad \boldsymbol{b} = -c_{\mathcal{E}}(\boldsymbol{x}_k)^T, \quad \boldsymbol{C} = \nabla c_{\mathcal{I}}(\boldsymbol{x}_k), \quad \boldsymbol{d} = -c_{\mathcal{I}}(\boldsymbol{x}_k)^T \tag{31}$$

where $\boldsymbol{B}_k$ is a BFGS approximation of the Hessian at the k'th iteration. With the individual matrices and vectors having shape $\boldsymbol{H}$ $(n \times n)$, $\boldsymbol{g}$ $(n \times 1)$, $\boldsymbol{A}$ $(m_e \times n)$, $\boldsymbol{b}$ $(m_e \times 1)$, $\boldsymbol{C}$ $(m_i \times n)$ and $\boldsymbol{d}$ $(m_i \times 1)$. The quadratic sub-problem will return a solution vector, $\boldsymbol{x}$ $(n \times 1)$, two sets of Lagrangian multipliers, $\boldsymbol{\lambda}_{\mathcal{E}}$ $(m_e \times 1)$, $\boldsymbol{\lambda}_{\mathcal{I}}$ $(m_i \times 1)$ and slack variables, $\boldsymbol{s}$ $(m_i \times 1)$.

Elastic Mode SQP

The formulation of the *elastic mode* problem is

$$\begin{aligned} \min_{x} \quad & f(\boldsymbol{x}) + \mu \sum_{i \in \mathcal{E}} (v_i + w_i) + \mu \sum_{i \in \mathcal{I}} t_i \\ \text{s.t.} \quad & c_i(\boldsymbol{x}) = v_i - w_i, \quad i \in \mathcal{E} \\ & c_i(\boldsymbol{x}) \geq -t_i, \quad i \in \mathcal{I} \\ & \boldsymbol{v},\, \boldsymbol{w},\, \boldsymbol{t} \geq 0 \end{aligned} \tag{32}$$

Combining (30) and (32) by replacing $f(x)$ with the Taylor expansion and re-arranging terms to match an algorithmic formulation

$$\begin{aligned} \min_{x} \quad & f_k + \nabla f_k^T \boldsymbol{p} + \frac{1}{2} \boldsymbol{p}^T \nabla^2_{xx} \mathcal{L}_k \boldsymbol{p} + \mu \sum_{i \in \mathcal{E}} (v_i + w_i) + \mu \sum_{i \in \mathcal{I}} t_i \\ \text{s.t.} \quad & \nabla c_i(\boldsymbol{x}_k)^T \boldsymbol{p} - v_i + w_i = -c_i(\boldsymbol{x}_k), \quad i \in \mathcal{E} \\ & \nabla c_i(\boldsymbol{x}_k)^T \boldsymbol{p} + t_i \geq -c_i(\boldsymbol{x}_k), \quad i \in \mathcal{I} \\ & \boldsymbol{v},\, \boldsymbol{w},\, \boldsymbol{t} \geq 0 \end{aligned}$$

Given the additional variables, $\boldsymbol{v}$, $\boldsymbol{w}$ and $\boldsymbol{t}$, the sub-problem is no longer of the same dimension as the original NLP. Let $\boldsymbol{p}_{sub}$ be the solution to the sub-problem, and $\boldsymbol{p}$ be the part relevant to the main problem. Then solution vector has the form

$$\boldsymbol{p}_{sub} = \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{v} \\ \boldsymbol{w} \\ \boldsymbol{t} \end{bmatrix}$$

where the shape of the individual vectors being $\boldsymbol{p}$ $(n \times 1)$, $\boldsymbol{v}$ $(m_e \times 1)$, $\boldsymbol{w}$ $(m_e \times 1)$ and $\boldsymbol{t}$ $(m_i \times 1)$ and consequently $\boldsymbol{p}_{sub}$ has the dimension $(n + 2m_e + m_i) \times 1$. On top of that the QP returns Lagrangian multipliers, $\boldsymbol{\lambda}_{\mathcal{E}}$ $(m_e \times 1)$, $\boldsymbol{\lambda}_{\mathcal{I}}$ $(m_i \times 1)$ and slack variables $\boldsymbol{s}$ $(m_i \times 1)$.

The linear algebra formulation of the problem is given as follows

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{B}_k & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}\epsilon & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}\epsilon & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}\epsilon \end{bmatrix}, \quad \boldsymbol{g} = \begin{bmatrix} \nabla f(\boldsymbol{x}_k)^T \\ \mu e_v^T \\ \mu e_w^T \\ \mu e_t^T \end{bmatrix}$$

where $\boldsymbol{B}$ is the Hessian of the main-problem or an approximation of it. The other the diagonal terms $\boldsymbol{I}\epsilon$ with $\epsilon \approx 10^{-6}$ in the Hessian matrix are added for numerical stability, but not actually part of the main formulation. The block matrix, $\boldsymbol{H}$ is of shape $(n + 2m_e + m_i) \times (n + 2m_e + m_i)$ and $\boldsymbol{g}$ is of shape $(n + 2m_e + m_i) \times 1$. The

equality constraints are given by

$$\boldsymbol{A} = \begin{bmatrix} \nabla c_{\mathcal{E}}(\boldsymbol{x}_k) & -\boldsymbol{I} & \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{b} = -c_{\mathcal{E}}(\boldsymbol{x}_k)^T$$

where $\boldsymbol{A}$ has shape $m_e \times (n + 2m_e + m_i)$ and $\boldsymbol{b}$ has shape $m_e \times 1$. Lastly the inequality constraints have the form

$$\boldsymbol{C} = \begin{bmatrix} \nabla c_{\mathcal{I}}(\boldsymbol{x}_k) & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{d} = \begin{bmatrix} -c_{\mathcal{I}}(\boldsymbol{x}_k)^T \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}$$

8.3.2. Interior-Point Method

8.4. CMA-ES Algorithms

8.4.1. Population-Based Algorithm

The population based CMA-ES algorithm is a Covariance Matrix Adaptive Evolution Strategy algorithm. The fundamental difference between CMA-ES and other ES' is that CMA-ES employs a cumulative learning from previous population samples in order to focus convergence towards areas of higher fitness. The algorithm used in `modpy` is a so-called $(\mu/\mu_w, \lambda)$-CMA-ES which states that the new candidate solution is a weighted combination of the μ best samples from the population λ.

The algorithm iteratively updates five state variables until a convergence criteria is met

- $\boldsymbol{m}_k \in \mathbb{R}^n$, the current candidate solution to the optimization problem
- $\sigma_k > 0$, the step-size control
- $\boldsymbol{C}_k$, a symmetric and positive-definite $n \times n$ covariance matrix
- $\boldsymbol{p}_\sigma \in \mathbb{R}^n$, the step-size evolution path
- $\boldsymbol{p}_c \in \mathbb{R}^n$, the covariance evolution path

where $\boldsymbol{m}_0$ and σ_0 are passed to the algorithm. $\boldsymbol{C}_0$ is usually initialized as the identity matrix unless prior knowledge is available. Similarly the evolution paths are initialized as 0-vectors.

Any iteration, k, is started by drawing $\lambda > 1$ samples from a multivariate normal-distribution $\mathcal{N}\left(\boldsymbol{m}_k, \sigma_k^2 \boldsymbol{C}_k\right)$

$$\boldsymbol{x}_i \sim \boldsymbol{m}_k + \sigma_k \cdot \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{C}_k\right)$$

the λ samples are then sorted based on their fitness. The fitness is evaluated using fitness function, $f(\cdot)$, which is passed to the algorithm. Let

$$\{\boldsymbol{x}_{i:\lambda} \mid i = 1, ..., \lambda\} = \{\boldsymbol{x}_i \mid f(\boldsymbol{x}_{1:\lambda}) \leq ... \leq f(\boldsymbol{x}_{\mu:\lambda}) \leq ... \leq f(\boldsymbol{x}_{\lambda:\lambda})\}$$

denote the sorted list of samples in ascending order of fitness. Then the $k+1$ candidate solution is the weighted combination of the μ fittest samples

$$\boldsymbol{m}_{k+1} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}$$

where the weights satisfy $w_1 \geq w_2 \geq ... \geq w_\mu \geq 0$ and

$$\sum_{i=1}^{\mu} w_i = 1$$

Subsequent to the calculation of the new candidate solution the various covariance parameters are updated. Step-size control, σ_k, is updated using the *cumulative step-size adaption*, also called the *path length control*. First the evolution path, p_σ, is updated

$$\boldsymbol{p}_\sigma \leftarrow (1-c_\sigma)\boldsymbol{p}_\sigma + \sqrt{\mu_w - \mu_w(1-c_\sigma)^2}\cdot \boldsymbol{C}_k^{-\frac{1}{2}} \cdot \frac{\boldsymbol{m}_{k+1}-\boldsymbol{m}_k}{\sigma_k}$$

where $c_\sigma > 1$ is the inverse of the backward time horizon, d_σ is a damping parameter and μ_w is called the variance effective selection mass and is defined as

$$\mu_w = \left(\sum_{i=1}^{\mu} w_i^2\right)^{-1}$$

then the step-size is updated

$$\sigma_{k+1} = \sigma_k \cdot \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\boldsymbol{p}_\sigma||}{\mathbb{E}||\mathcal{N}(0,I)||} - 1\right)\right)$$

the step-size is increased if and only the norm of $\boldsymbol{p}_\sigma$ is larger than the expected value

$$\mathbb{E}||\mathcal{N}(0,I)|| = \sqrt{2}\cdot\frac{\Gamma\left(\frac{n+1}{2}\right)}{\Gamma\left(\frac{n}{2}\right)} \approx \sqrt{n}\left(1-\frac{1}{4n}+\frac{1}{21n^2}\right)$$

and decreased if it is smaller. Finally the covariance matrix is updated. Again, the evolution path is updated first

$$\boldsymbol{p}_c \leftarrow (1-c_c)\boldsymbol{p}_c + \mathbb{1}_{[0,\alpha\sqrt{n}]}\left(||\boldsymbol{p}_\sigma||\right)\sqrt{\mu_w - \mu_w(1-c_\sigma)^2}\cdot\frac{\boldsymbol{m}_{k+1}-\boldsymbol{m}_k}{\sigma_k}$$

where $\mathbb{1}_{[0,\alpha\sqrt{n}]}\left(||p_\sigma||\right)$ is the indicator function

$$\mathbb{1}_{[0,\alpha\sqrt{n}]}\left(||\boldsymbol{p}_\sigma||\right) = \begin{cases} 1, & \text{if } ||\boldsymbol{p}_\sigma|| \in [0,\alpha\sqrt{n}] \\ 0, & \text{otherwise} \end{cases}$$

and then the covariance matrix

$$\boldsymbol{C}_{k+1} = (1-c_1-c_\mu+c_s)\boldsymbol{C}_k + c_1\boldsymbol{p}_c\boldsymbol{p}_c^T + \frac{c_\mu}{\sigma_k^2}\sum_{i=1}^{\mu} w_i\left(\boldsymbol{x}_{i:\lambda}-\boldsymbol{m}_k\right)\left(\boldsymbol{x}_{i:\lambda}-\boldsymbol{m}_k\right)^T$$

the parameter, c_s, makes up for a small variance loss in case the indicator function is zero

$$c_s = (1-\mathbb{1}_{[0,\alpha\sqrt{n}]}\left(||\boldsymbol{p}_\sigma||\right))c_1c_c(2-c_c)$$

the remainder of the parameters are either supplied to the algorithm or calculated based on some heuristics as a function of the problem dimension. The heuristics are defined as

$$
\begin{aligned}
\lambda &= 4 + \lfloor 3\ln(n) \rfloor \\
\mu &= \left\lfloor \frac{\lambda}{2} \right\rfloor \\
w_i &= \ln\left(\mu + \frac{1}{2}\right) - \ln(i), \quad i = 1, ..., \mu \\
c_c &= \frac{4\frac{\mu_{eff}}{n}}{N + 4 + 2\frac{\mu_{eff}}{n}} \\
c_\sigma &= \frac{\mu_{eff} + 2}{n + \mu_{eff} + 5} \\
c_1 &= \frac{2}{(n + 1.3)^2 + \mu_{eff}} \\
c_\mu &= \min\left(1 - c_1, 2 \cdot \frac{\mu_{eff} - 2 + \frac{1}{\mu_{eff}}}{(N+2)^2 + \mu_{eff}}\right) \\
d_\sigma &= 1 + 2 \cdot \max\left(0, \sqrt{\frac{\mu_{eff} - 1}{N + 1}} - 1\right) + c_\sigma
\end{aligned}
$$

8.5. Increasing Population

The IPOP-CMA-ES algorithm is an increasing population size CMA-ES. It wraps the CMA-ES outlined Section 8.4.1 in an outer loop, which increases the initial standard deviation, σ_0, and population size, λ, in every iteration that the basic CMA-ES fails to converge. Dependent on the reason for diverging the two variables are increased with variable factors.

8.5.1. Elitist Algorithm

As concluded in [10] the (1+1)-CMA-ES is a relatively capable EA for solving unimodal constrained optimization problems. It is however a poor global optimizer and consequently is not useful for multimodal problems.

8.6. Hybrid Algorithms

Hybrid algorithms are combinations of evolution strategy algorithms and gradient-based algorithms. The logic is get close to the global optimum using an evolution strategy, to some tolerance ϵ_{ES}, and then use the found optimum $\boldsymbol{x}^*_{ES}$ as start-guess for the gradient-based algorithm to refine it to the desired tolerance, ϵ, with the final result, $\boldsymbol{x}^*$.

8.7. Multi-Modal Optimization Algorithms

Multi-Modal Optimization (MMO) is the purpose of finding not only a single global optimum, but rather all optimums (local and global). This is achieved through the method *niching*, which essentially splits the domain into different areas which are assumed to contain an optimum each. Within a given niche, a normal (single) optimization algorithm is run to find the optimum within that niche. Thus an MMO algorithm is characterized by the method used to split the domain into niches and the optimization algorithm run to find the optimum in those niches. `modpy` uses a niching technique known as Hill-Valley Clustering [15].

9. Bayesian Optimization

Bayesian Optimization is an optimization method which is particularly well suited for black-box optimization with expensive function evaluations.

See description in Rudholm, J., Wojciechowski, A. (2007). *A Method for Simulation Based Optimization Using Radial Basis Functions.* (http://www.math.chalmers.se/ mipat/LATEX/JohanAdam.pdf) for inspiration.

9.1. Acquisition Functions

The acquisition function is used as the objective function for proposal of new function calls. It is designed to weight the need for exploration of unknown areas of the domain versus refining areas of interest.

Expected Improvement. This acquisition function maximizes the expected improvement over the current best estimate, given by a closed form Gaussian process:

$$a_{EI}(\boldsymbol{\theta}) = (\mu(\boldsymbol{\theta}) - f_{\max}) \cdot \Phi\left(\frac{\mu(\boldsymbol{\theta}) - f_{\max} - \xi}{\sigma(\boldsymbol{\theta})}\right) + \sigma(\boldsymbol{\theta}) \cdot \phi\left(\frac{\mu(\boldsymbol{\theta}) - f_{\max} - \xi}{\sigma(\boldsymbol{\theta})}\right)$$

where $\mu(\boldsymbol{\theta})$ and $\sigma(\boldsymbol{\theta})$ are the mean and standard deviation from Gaussian process (the proxy-model), $f_{\max}$ is the current best estimate, from a previous sample point, ξ is a trade-off parameter between exploration and refinement. The functions $\phi(\cdot)$ and $\Phi(\cdot)$ are the probability- and cumulative density functions and $\boldsymbol{\theta}$ is the parameter being optimized.

Probability of Improvement. This acquisition function is a simpler version of the Expected Improvement, attempting to maximize the probability instead of the expected value.

$$a_{PoI}(\boldsymbol{\theta}) = \Phi\left(\frac{\mu(\boldsymbol{\theta}) - f_{\max} - \xi}{\sigma(\boldsymbol{\theta})}\right)$$

where the various input is similar to that of expected improvement.

Upper Confidence Bound. An alternative to the two other approaches is to instead consider the confidence bounds. This approach puts a higher emphasis on exploring areas far away from previously sampled points.

$$a_{UCB}(\boldsymbol{\theta}) = \mu(\boldsymbol{\theta}) + \kappa \cdot \sigma(\boldsymbol{\theta})$$

where $\kappa > 0$ is a trade-off parameter, corresponding to confidence intervals of a normal distribution.

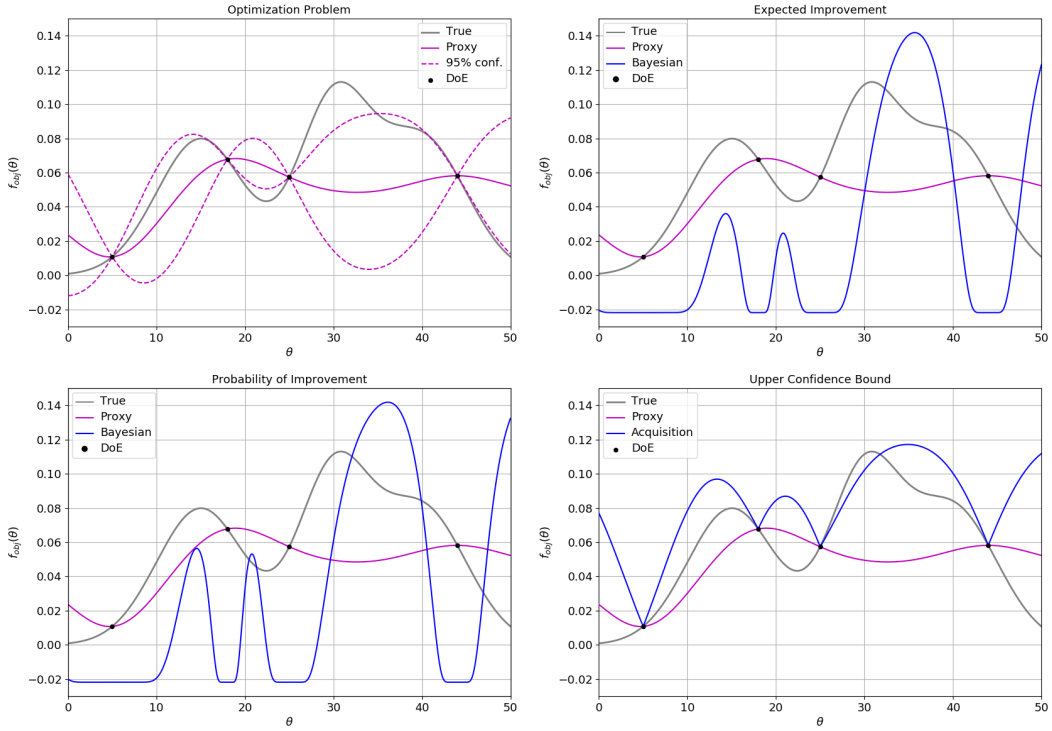


Figure 20: Example of acquisition functions in 1D. For EI and PoI the trade-off parameter is $\xi = 0.01$ for UCB the trade-off parameter is $\kappa = 3$. The values EI and PoI does not correspond to the y-axis on the plots, but are hidden as the order of magnitude is irrelevant.

Fig. 20 shows the different behaviour of the acquisition functions. For this simple example EI and PoI have a similar shape, but different values. All three seem to attain their maximum value at the same position, around $\theta = 35$, but UCB differs slightly from the others by having a less accentuated peaks.

The behaviour of EI and PoI is not strictly convex and consequently cannot be solved by a gradient-based optimization algorithm. UCB is piece-wise convex, but also only piece-wise differentiable, with issues occurring in the existing sample points.

Due to the multi-modal behaviour of the acquisition functions, Bayesian Optimization can be solved either sequentially or in parallel.

9.2. Sequential

In Sequential Bayesian Optimization the acquisition function is optimized to find the global optimum. The parameter at the global optimum is passed to the simulator, a new point is added to the proxy-model which is updated and the process is repeated. In the case of Fig. 20 the first iteration would propose $\theta = 35$ as the new point for simulation.

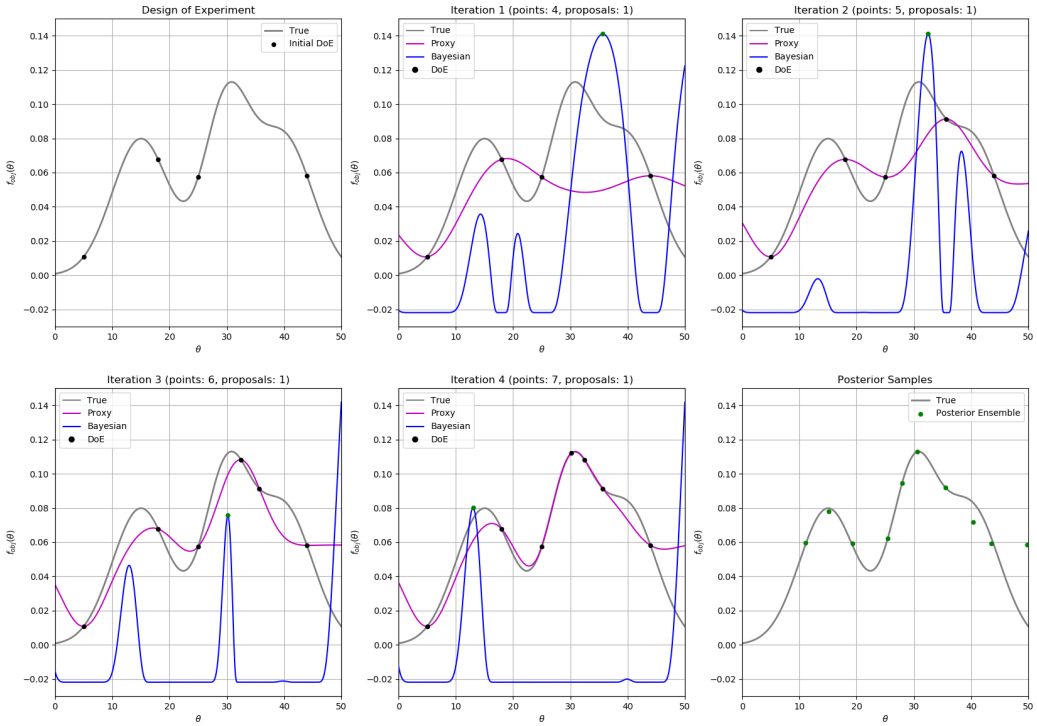


Figure 21: Example of Sequential Bayesian Optimization in 1D. Plot 1: True underlying objective function and a poorly chosen initial experimental design. Plot 2-5: Iterative updates of the proxy-model based on Expected Improvement. Plot 6: Samples from the posterior distribution sampled based on the last iteration of the proxy-model.

Fig. 21 shows how the proxy-model is sequentially refined by proposing new function calls to the true objective function. Notice that this example is not evolved optimally as a local optimum is found in the 3^{rd} and 4^{th} iteration.

9.3. Parallel

Parallel Bayesian Optimization differs from Sequential Bayesian Optimization by proposing multiple new functions calls in each iteration. While all the local optimums in theory provide less information than the global one, it can be convenient to include them when the function calls of the underlying problem can be launched in parallel.

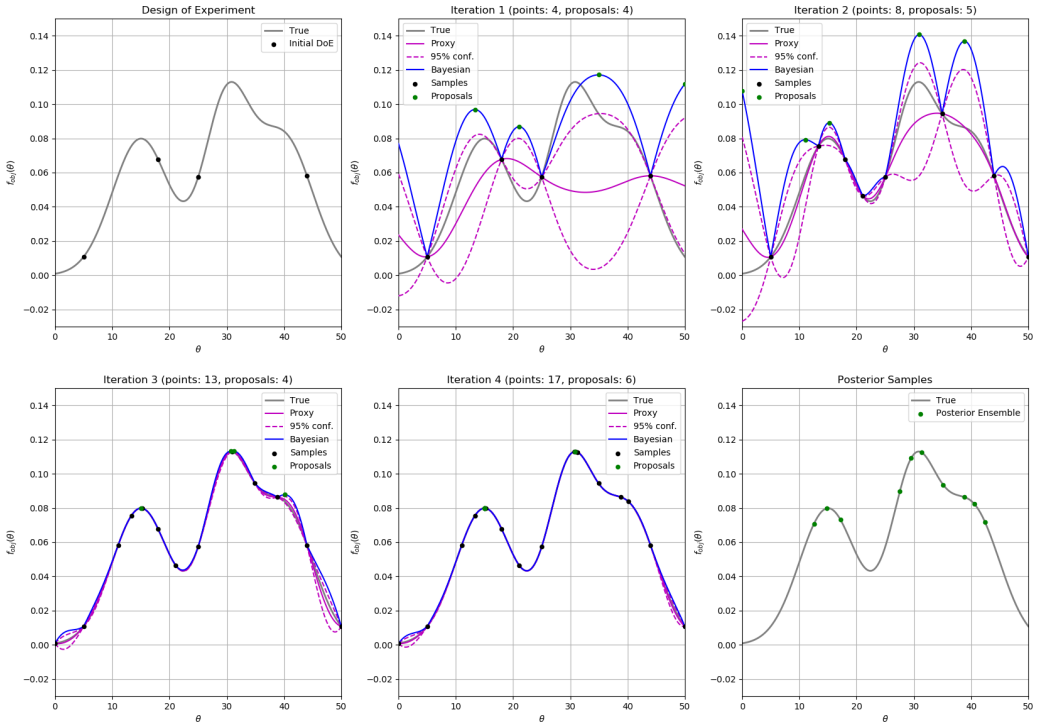


Figure 22: Example of Parallel Bayesian Optimization in 1D. Plot 1: True underlying objective function and a poorly chosen initial experimental design. Plot 2-5: Iterative updates of the proxy-model based on Upper Confidence Bound. Plot 6: Samples from the posterior distribution sampled based on the last iteration of the proxy-model.

In Fig. 22 the proxy-model is refined quicker than what is shown in Fig. 21, but at the cost of additional calls to the true objective function. To limit the number of runs per iteration in highly multi-modal problems, a threshold can be set to only propose local optimums for which $f(\boldsymbol{\theta}_{\text{local}}) \geq \epsilon f(\boldsymbol{\theta}_{\text{global}})$ where $\theta \in [0, 1]$. In general it cannot be confirmed that a given $\boldsymbol{\theta} = \boldsymbol{\theta}_{\text{global}}$ so instead $\boldsymbol{\theta}_{\text{local}}$ is compared to the best solution found.

Whether to use a sequential or parallel approach depends entirely on the difficulties of solving the true problem. The sequential approach will require more iterations, but uses only the points which provide maximum information. The parallel approach requires less iterations, but will include points of less information. If the true problem can be solved in parallel, e.g. if the problem is solved on a HPC cluster, then the parallel approach is the fastest. If only one problem can be solved at a time, the sequential approach is the best.

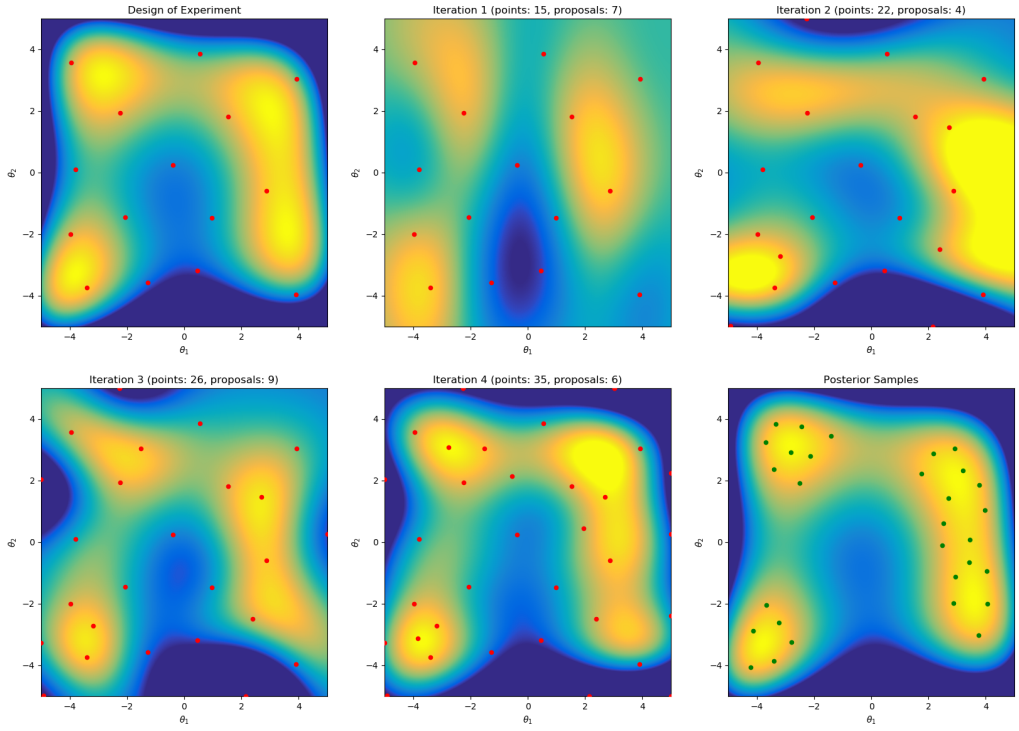


Figure 23: Example of Parallel Bayesian Optimization in 2D. Plot 1: True underlying objective function and the initial experimental design. Plot 2-5: Iterative updates of the proxy-model based on Upper Confidence Bound. Plot 6: Samples from the posterior distribution sampled based on the last iteration of the proxy-model.

9.4. Improvements

Bayesian Optimization balances the objective of exploring the uncertain parts of the domain and refining in areas of higher interest. As uncertainty is reduced during the iterative phase, the trade-off turns more and more towards areas of high likelihood. This can lead to clustering of points where proposals are close to existing points as seen in iteration 3 and 4 on Fig. 22. This is fine when the objective is to find a single deterministic optimum, but undesirable when the primary objective is to refine the proxy-model. To further improve the investigation of the underlying objective function and avoid clustering of simulation points, a constraint can be added for each existing point

$$\sum_{i=1}^{d} \left(\theta_{n+1}^{(i)} - \theta_{j}^{(i)} \right)^2 \geq r^2 \quad j = 1, ..., n \tag{33}$$

where d is the dimension, $\theta_{n+1}^{(i)}$ is the i'th index of the proposed point, $\theta_{j}^{(i)}$ is the i'th index of the j'th existing point and n is the number of existing points. In 2D this corresponds to the circle equation and in 3D to the sphere equation. This ensures that each proposed point is at some fixed distance, r, away from existing points. For parallel Bayesian optimization this does not take into account clustering of proposed points in the same iteration. The use of niching algorithms should account for this problem though. Fig. 24 shows the same 1D example as used previously, but with the added constraint.

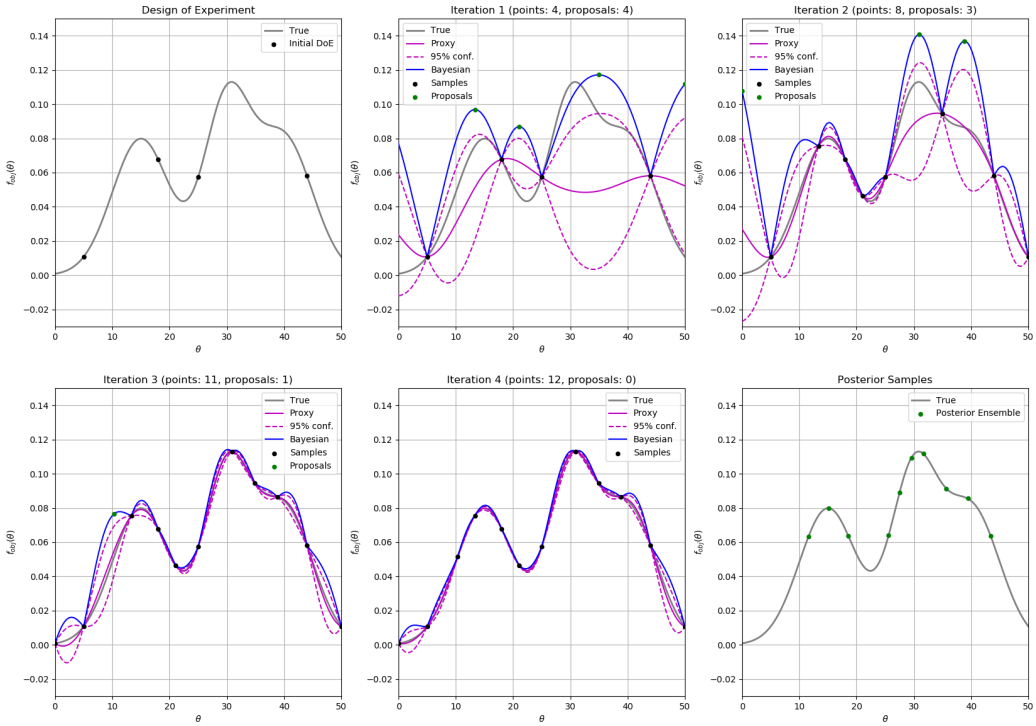


Figure 24: Example of Parallel Bayesian Optimization in 1D including the constraints of (33). Plot 1: True underlying objective function and a poorly chosen initial experimental design. Plot 2-5: Iterative updates of the proxy-model based on Upper Confidence Bound. Plot 6: Samples from the posterior distribution sampled based on the last iteration of the proxy-model.

As seen from Fig. 22 and Fig. 24 the added constraint changes the proposed samples to reduce clustering of samples in already investigated areas. It is clear this results in a proxy-model that is slightly more uncertain. This added uncertainty seems minor compared to the reduction of simulations, from 23 in the unconstrained case to 12 in the constrained, i.e. a reduction of half.

10. Posterior Distribution

The likelihood function (17) introduced earlier is actually a special case of the posterior probability density function, $f(\theta \mid x)$, with uniform priors

$$f(\theta \mid x) = \frac{f(x \mid \theta)}{f(x)} f(\theta) \tag{34}$$

where $f(\theta \mid x)$ is the posterior probability, $f(x \mid \theta)$ is the probability density function of X given θ, $f(x)$ is a normalization constant and $f(\theta)$ is the a priori probability. The normalization constant is given by

$$f(x) = \int_{\Theta} f(x \mid \theta) f(\theta) \, \mathrm{d}\theta \tag{35}$$

where Θ is the domain of θ. $f(x)$ ensures that the integral of the posterior distribution integrates to unity. The integral of (35) is not readily available. Luckily algorithms are available that allow to sample from the posterior distribution without determining (35). A well known option is the Metropolis-Hastings algorithm which relies on Markov Chain Monte Carlo (MCMC).

10.1. Markov Chain Monte Carlo

10.1.1. Markov Chain

A Markov Chain is a stochastic model which follows a Markov Process. This means that predictions regarding the future state depend solely on the current state. A Markov process can either be in discrete time or continuous time with an integer Random Walk being an example of a discrete time process and a Wiener process (Brownian motion) being an example of a continuous time process.

A discrete-time Markov chain is a sequence of random variables $\boldsymbol{X} = (X_1, X_2, ..., X_n)$ which follows a Markov process

$$\Pr\left[X_{n+1} = x \mid X_1 = x_1, X_2 = x_2, ..., X_n = x_n\right] = \Pr\left[X_{n+1} = x \mid X_n = x_n\right]$$

the possible values x_i of X_i form a countable set, $\mathcal{S}$ which is called the state-space of the chain.

10.1.2. Sampling

Markov Chain Monte Carlo methods comprise a class of algorithms which allows sampling from probability distributions. This is particular useful when the closed-form solution of the probability distribution is unknown. The sampling occurs by constructing a Markov chain which has the desired probability distribution as its equilibrium distribution. The goal of the sampling is to estimate the mean of some distribution. Disregard for the moment the notion of the posterior probability distribution, and consider the mean of some arbitrary function, $f(x)$, which follows the distribution $\pi(x)$

$$\mathbb{E}[f] = \int f(x)\pi(x)\ \mathrm{d}x$$

assuming a sequence of parameters, $X = (x_1, x_2, ..., x_n)$ has been sampled from some distribution with probability density function, $\pi(x)$, then the mean may be approximated by

$$\mathbb{E}[f] \approx \bar{f} = \frac{1}{n}\sum_{i=1}^{n} f(x_i) \tag{36}$$

The strong law of large numbers states

$$\lim_{n\to\infty} \bar{f} = \mathbb{E}[f]$$

consequently for any, $n < \infty$ there will be some errors related to the estimate. An estimate of the errors of $\bar{f}$ can be obtained by estimating $\bar{f}$ using (36) m times. Let $\sigma^2_{\bar{f}}$ denote the variance of the m estimates of $\bar{f}$. The central limit theorem states

$$\sqrt{m}\left(\bar{f} - \mathbb{E}[f]\right) \sim \mathcal{N}(0, \sigma^2) \Leftrightarrow \lim_{m\to\infty} \sigma^2_{\bar{f}} = \frac{\sigma^2}{m}, \quad \mathrm{Var}[\bar{f}] = \sigma^2$$

logically, σ^2 will depend on the MCMC method. For instance if the length of the MCMC chain, n, is short, then σ^2 will be large. It can be shown that

$$\sigma^2 = \tau_f \sigma^2_{\bar{f}}$$

where τ_f is the auto-correlation time, i.e. the number of samples required to consider two samples x_i and $x_{i+\tau_f}$ independent. This value is defined as

$$\tau_f = \sum_{t=-\infty}^{\infty} \rho_t = 1 + 2\sum_{t=1}^{\infty} \rho_t$$

where ρ_t is the auto-correlation at at lag t. Presumably it should be possible to truncate this sum after n, unfortunately the estimates of ρ_t are noisy and consequently are not guaranteed to tend to 0 as $t \to \infty$. τ_f is considered a measure of the quality of the MCMC, with low values indicating a fast rate of convergence for the chain and vice versa for high values. The auto-correlation time can be used to quantify the effective-sample size (ESS)

$$n_{\text{eff}} = \frac{n}{\tau_f} \tag{37}$$

where n is the total number of samples and n_{eff} is the number of samples which can be considered independent. E.g. a chain of $n = 10^4$ steps with an auto-correlation time of $\tau_f = 50$ only has 200 independent samples.

10.1.3. Auto-Correlation

Auto-correlation is the correlation of a random process, X, with itself after some shift in the sequence. This is often related to time-series analysis, and is used to measure the degree correlation between two points of the process X_{t_1} and X_{t_2}

$$\rho_{XX}(t_1, t_2) = \frac{K_{XX}(t_1, t_2)}{\sigma_{t_1}\sigma_{t_2}} = \frac{\mathbb{E}\left[(X_{t_1} - \mu_{t_1})\overline{(X_{t_2} - \mu_{t_2})}\right]}{\sigma_{t_1}\sigma_{t_2}}$$

where $\rho_{XX} \in [-1, 1]$ is the autocorrelation, $K_{XX} \in \mathbb{R}$ is the auto-covariance, the bar denotes the complex conjugate and μ_{t_1}, σ_{t_1} and μ_{t_2}, σ_{t_2} is the mean and standard deviation of X_{t_1} and X_{t_2} respectively. Often the process can be assumed to be wide-sense stationary in which case the auto-correlation depends not on the specific times, t_1 and t_2 but rather the distance between them, $\tau = t_2 - t_1$ (known as the lag-time)

$$\rho_{XX}(\tau) = \frac{K_{XX}(\tau)}{\sigma^2} = \frac{\mathbb{E}\left[(X_t - \mu)\overline{(X_{t+\tau} - \mu)}\right]}{\sigma^2}$$

While a Markov chain is not a time-series process, the way it sequentially samples a step x_{n+1} from x_n introduces a high degree of auto-correlation in the process. Methods such as Hamiltonian Monte Carlo attempts to overcome this by taking steps in a smarter way than the Metropolis-Hastings algorithm, leading to less lag-time and faster convergence.

10.1.4. Monte Carlo

The combination of Monte Carlo sampling and Markov chains comes from the process of randomly guessing the next state of the process, by some addition to the current state

$$x_{n+1} = x_n + \Delta, \quad \Delta \sim \mathcal{D} \tag{38}$$

where Δ is some step which is sampled from some distribution, $\mathcal{D}$. During the sampling of the chain, the next step, x_{n+1} is either accepted or rejected based on whether this next step is an improvement in the likelihood. The different available algorithms such as Metropolis-Hastings and Hamiltonian Monte Carlo all use the same fundamental approach as outlined in (38), but differ in the way they propose and accept/reject the change, Δ. The sequence (or chain) of sampled variables, $X = \{x_1, x_2, .., x_n\}$ will converge towards the equilibrium distribution as $n \to \infty$.

10.1.5. Thinning

Thinning the Markov chain is the act of selecting a sub-sample which can be assumed statistically independent (uncorrelated). This can be achieved by selecting every τ sample, where τ can either be an arbitrary number of sufficient size or the estimated auto-correlation time.

$$\mathcal{S}_\tau = \{x_i \in \mathcal{S} \mid i = 1, \tau, 2 \cdot \tau, ..., N\}$$

where $\mathcal{S}$ is the state-space of the chain. The thinned chain has a length equal to the effect sample size, n_{eff}. When MCMC is used for parameter estimation (approximating an integral) it is generally ill-advised to thin the chain as this results in a larger variance of the estimate [14]. This point can even be shown analytically for simple problems. However, as also mentioned in [14] there can be good reasons for thinning the chain anyway. In fact for the purpose of `modpy` the chains will typically be thinned and the rationale for this will be explained later when discussing representative sub-sampling.

10.2. Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm is used to sample from an arbitrary probability distribution $f(x)$ as long as we know some function, $g(x)$, which is proportional to it

$$f(x) = k \cdot g(x) \Leftrightarrow f(x) \propto g(x)$$

the fact that it only requires a function which is proportional to $f(x)$ is the key to its usefulness.

Example 10.1 (Normal Distribution)**.**
Consider the known formula for the the normal distribution

$$f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} = k \cdot g(x;\mu,\sigma)$$

where

$$k = \frac{1}{\sqrt{2\pi}\sigma}, \quad g(x;\mu,\sigma) = e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

This separation is possible because the normalization factor, k, is independent of x and hence $g(x;\mu,\sigma) \propto f(x;\mu,\sigma)$.

Example 10.2 (Posterior Distribution)**.**
Another example which is of primary importance is its relationship to the calculation of the posterior probability (34), namely it is evident that

$$f(\theta \mid x) = \frac{f(x \mid \theta)}{f(x)} f(\theta) = k \cdot f(x \mid \theta) f(\theta)$$

where

$$k = \frac{1}{f(x)}, \quad g(\theta \mid x) = f(x \mid \theta) f(\theta)$$

similar to the example of the normal distribution, the normalization factor, $f(x)$, is independent of the variable, θ. Consequently, it is possible to draw samples from the posterior probability, $f(\theta \mid x)$, by knowing only the prior probability, $f(\theta)$, and the likelihood, $f(x \mid \theta)$.

In practice the algorithm uses the log-likelihood instead of the likelihood to avoid numerical underflow.

10.3. Hamiltonian Monte Carlo

The implementation is based on [13].

10.4. Representative Sub-sampling

MCMC algorithms such as Metropolis-Hastings and Hamiltonian Monte Carlo are - as mentioned previously - used to estimate the expected value (integral) of a posterior distribution. They are not as such used to perform random sampling from the posterior distribution. As part of the estimation process the state-space of the chain, $\mathcal{S}$, is obtained, providing a countable set of samples from the posterior distribution. The size of this set will typically be too large to be feasible for simulation, consequently a representative sub-set has to be

selected. Logically this sub-set should span the posterior distribution to some given threshold of probability, retaining as much information about the state-space with as few samples as possible.

10.4.1. Heuristic

After thorough review, no obvious approaches seem to be available in literature for solving the aforementioned problem. Consequently a heuristic approach is used. As mentioned there is two key objectives

- Span the posterior distribution to some given threshold of probability.
- Retain maximum information with as few samples as possible.

The first point can be achieved by selecting a sub-set of the thinned state-space $\mathcal{S}_\tau$ for which the probability of $x \in \mathcal{S}$ is above a given threshold, $\alpha \in [0, 1]$

$$\mathcal{S}_\alpha = \{x \in \mathcal{S}_\tau \mid F(x) \geq \alpha\}$$

where $F(x)$ is the cumulative density function of the posterior distribution. For a countable set this can be achieved by sorting the set in ascending order, finding the α'th percentile and then retaining all samples which have a likelihood above or equal to that percentile. The rationale for using the thinned state-space is that the size of posterior samples is much lower than the length of the chain, so it makes sense to only sub-sample from the most representative initial samples. Notice that for this to work on a countable set, the chain should first be reduced to only include values which can be assumed greater than zero.

The second objective can be achieved by the use of an experimental design, more specifically a space-filling design. By using $\mathcal{S}_\alpha$ as the initial set of points, these designs can perform a sub-sampling which avoids clustering by use of max-min distance approach.

This heuristic does not provide any mathematical guarantees that the posterior distribution will be representatively sampled, but is based on logic and can be shown to yield go results on simple problems. TODO!

11. Optimization under Uncertainty

12. Model Predictive Control

Part III
Appendix

13. Appendix: Special Functions

13.1. Special Functions

13.1.1. Error Function

The error function is the finite integral

$$\mathrm{Erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \, \mathrm{d}t \tag{39}$$

13.1.2. Gamma Function

The gamma function is the infinite integral

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} \mathrm{d}t \tag{40}$$

which also comes in the form of two finite integrals, namely the lower and upper incomplete gamma function

$$\gamma(x; s) = \int_0^x t^{s-1} e^{-t} \mathrm{d}t \tag{41}$$

$$\Gamma(x; s) = \int_x^\infty t^{s-1} e^{-t} \mathrm{d}t \tag{42}$$

with the relationship

$$\gamma(x; s) + \Gamma(x; s) = \Gamma(s) \tag{43}$$

for any given x. There exists no closed-form analytical solution to the inverse of the gamma function. Instead numerical algorithms are used to approximate it.

13.1.3. Beta Function

The beta function is the finite integral

$$\mathrm{B}(\alpha, \beta) = \int_0^1 t^{\alpha-1} (1-t)^{\beta-1} \mathrm{d}t \tag{44}$$

for $\mathrm{Re}(\alpha) > 0$ and $\mathrm{Re}(\beta) > 0$. It is related to the gamma function in the following way

$$\mathrm{B}(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha + \beta)} \tag{45}$$

Similar to the gamma function this also comes in the form of a lower incomplete beta function given by

$$\mathrm{B}(x; \alpha, \beta) = \int_0^x t^{\alpha-1} (1-t)^{\beta-1} \mathrm{d}t \tag{46}$$

with the additional constraint $x \in [0, 1]$. The beta function also comes in a regularized form $I_x(\alpha, \beta)$, which is called the regularized lower incomplete beta function

$$I_x(\alpha, \beta) = \frac{\mathrm{B}(x; \alpha, \beta)}{\mathrm{B}(\alpha, \beta)} \tag{47}$$

There exists no closed-form analytical solution to the inverse of the beta function. Instead numerical algorithms are used to approximate it.

14. Appendix: Proofs

14.1. Domain Transform

The transforms referred to by a domain transform is either linear, logarithmic or a root transform of a given domain $z \in [a, b]$. The goal is to achieve a value $x = f(z) \in [a, b]$ which is a monotone transform of z with the same compact support as the original variable.

Linear Transform

The linear case is simply a one-to-one transform

$$x = f(z) = a + \frac{z-a}{b-a}(b-a) = z$$

Logarithmic Transform
A logarithmic transform with base n has the form

$$x = f(z) = n^{g(z)}$$

where $g(\cdot)$ is some transform of z different from $f(\cdot)$. In order for both z and x to have the same compact support the function $g(\cdot)$ has to be defined in a specific way. First define the normalized variable $y \in [0,1]$

$$y = \frac{z-a}{b-a}, \quad x \in [a,b]$$

then the logarithmic transform becomes

$$x = n^{\log_n(a)+\frac{z-a}{b-a}(\log_n(b)-\log_n(a))} = n^{\log_n(a)+y\log_n\left(\frac{b}{a}\right)} = n^{\log_n(a)} \cdot n^{y\log_n\left(\frac{b}{a}\right)} = a \cdot n^{\log_n\left(\left(\frac{b}{a}\right)^y\right)} = a \cdot \left(\frac{b}{a}\right)^y$$

where

$$g(z) = \log_n(a) + \frac{z-a}{b-a}\left(\log_n(b) - \log_n(a)\right)$$

as can be seen the transform is non-linear and independent of the logarithmic base, n.

Root Transform
A root transforms with root n is given by

$$x = f(z) = g(z)^n$$

as for the logarithmic case this requires that $g(\cdot)$ is defined in a specific way

$$x = \left(\sqrt[n]{a} + \frac{z-a}{b-a}\left(\sqrt[n]{b} - \sqrt[n]{a}\right)\right)^n = \left(\sqrt[n]{a} + y\left(\sqrt[n]{b} - \sqrt[n]{a}\right)\right)^n = \prod_{i=1}^{n}\left(\sqrt[n]{a} + y\left(\sqrt[n]{b} - \sqrt[n]{a}\right)\right)$$

where $g(\cdot)$ is given by

$$g(z) = \sqrt[n]{a} + \frac{z-a}{b-a}\left(\sqrt[n]{b} - \sqrt[n]{a}\right)$$

this is similar to the log-transform a non-linear transform, but clearly not independent of the choice of root.

Notice that in all three cases both $z \in [a,b]$ and $x \in [a,b]$. This trivial to see in the linear case and for the logarithmic and root one it can be shown by setting $z = a \Leftrightarrow y = 0$ and $z = b \Leftrightarrow y = 1$. For the logarithmic case

$$\text{Logarithmic } (z=a): \quad a \cdot \left(\frac{b}{a}\right)^0 = a \cdot 1 = a$$
$$\text{Logarithmic } (z=b): \quad a \cdot \left(\frac{b}{a}\right)^1 = a\frac{b}{a} = b$$

and for the root transform

$$\text{Root } (z=a): \quad \left(\sqrt[n]{a} + 0\left(\sqrt[n]{b} - \sqrt[n]{a}\right)\right)^n = \left(\sqrt[n]{a}\right)^n = a$$
$$\text{Root } (z=b): \quad \left(\sqrt[n]{a} + 1\left(\sqrt[n]{b} - \sqrt[n]{a}\right)\right)^n = \left(\sqrt[n]{a} + \left(\sqrt[n]{b} - \sqrt[n]{a}\right)\right)^n = \left(\sqrt[n]{b}\right)^n = b$$

hence the compact support of the domain is the same. Note that this kind of domain transform is the same as assuming the transformed variable, z follows a uniform distribution $z \sim U(a,b)$ and deriving a transform

on the basis of that.

14.2. Logarithmic Transform of Distributions

For any distribution, $X = n^Z$ for which Z follows a well-defined distribution and $n > 1$ is the logarithmic base, the PDF of X may be derived as follows

$$\begin{aligned} f_X(x) &= \frac{\mathrm{d}}{\mathrm{d}x}\Pr[X \leq x] = \frac{\mathrm{d}}{\mathrm{d}x}\Pr\left[\log_n(X) \leq \log_n(x)\right] = \frac{\mathrm{d}}{\mathrm{d}x}F_Z(\log_n(x)) = f_Z(\log_n(x))\frac{\mathrm{d}}{\mathrm{d}x}\log_n(x) \\ &= f_Z(\log_n(x))\frac{1}{x\ln(n)} \end{aligned} \tag{48}$$

when the logarithmic base is the natural logarithm this expression simplifies to

$$f_X(x) = f_Z\left(\ln(x)\right) \cdot \frac{1}{x} \tag{49}$$

14.3. Root Transform of Distributions

For any distribution, $X = Z^n$ for which Z follows a well-defined distribution and $n \in \mathbb{N}_+$ is the exponent, the PDF of X may be derived as follows

$$\begin{aligned} f_X(x) &= \frac{\mathrm{d}}{\mathrm{d}x}\Pr[X \leq x] = \frac{\mathrm{d}}{\mathrm{d}x}\Pr\left[\sqrt[n]{X} \leq \sqrt[n]{x}\right] = \frac{\mathrm{d}}{\mathrm{d}x}F_Z\left(\sqrt[n]{x}\right) = f_Z\left(\sqrt[n]{x}\right)\frac{\mathrm{d}}{\mathrm{d}x}\sqrt[n]{x} \\ &= f_Z\left(\sqrt[n]{x}\right) \cdot \frac{1}{n}x^{\frac{1}{n}-1} \end{aligned} \tag{50}$$

14.4. Parameters of the Log-Normal Distribution

Given a log-normally distributed variable

$$X = n^{\mu+\sigma Z} \tag{51}$$

where $Z \sim \mathcal{N}(0,1)$. Notice that the moment generating function of the normal distribution (with e substituted for an arbitrary base n) is

$$\mathrm{E}\left[n^{tY}\right] = n^{\mu t + \frac{1}{2}\sigma^2 t^2}, \quad Y \sim \mathcal{N}\left(\mu, \sigma^2\right), \quad t \in \mathbb{R}$$

from that it follows directly that

$$\mathrm{E}\left[X^t\right] = \mathrm{E}\left[n^{tY}\right]$$

and consequently the mean and variance of the log-normal distribution is defined by the moment generating function of its underlying normal distribution, namely

$$\mathrm{E}[X] = n^{\mu+\frac{1}{2}\sigma^2} \tag{52}$$

and

$$\begin{aligned} \mathrm{Var}[X] &= \mathrm{E}\left[\left(X - \mathrm{E}[X]\right)^2\right] = \mathrm{E}\left[X^2\right] - \mathrm{E}[X]^2 = \\ &= n^{2\mu+\frac{1}{2}\sigma^2 2^2} - \left(n^{\mu+\frac{1}{2}\sigma^2}\right)^2 = n^{2\left(\mu+\sigma^2\right)} - n^{2\mu+\sigma^2} = \left(n^{\sigma^2}-1\right)n^{2\mu+\sigma^2} \end{aligned}$$

This provides two equations with two unknowns. First isolating μ in the first one

$$\mu = \log_n\left(\mathrm{E}[X]\right) - \frac{1}{2}\sigma^2$$

inserting that into the second equation yields

$$\begin{aligned} \log_n\left(\mathrm{Var}[X]\right) &= \log_n\left(n^{\sigma^2}-1\right) + 2\mu + \sigma^2 = \log_n\left(n^{\sigma^2}-1\right) + 2\left(\log_n\left(\mathrm{E}[X]\right) - \frac{1}{2}\sigma^2\right) + \sigma^2 \\ &= \log_n\left(n^{\sigma^2}-1\right) + 2\log_n\left(\mathrm{E}[X]\right) \end{aligned}$$

solving this expression for σ^2 yields

$$\sigma^2 = \log_n\left(1 + \frac{\mathrm{Var}[X]}{\mathrm{E}[X]^2}\right)$$

Substituting this back into the expression for μ gives

$$\mu = \log_n(\mathrm{E}[X]) - \frac{1}{2}\sigma^2 = \log_n(\mathrm{E}[X]) - \frac{1}{2}\log_n\left(1 + \frac{\mathrm{Var}[X]}{\mathrm{E}[X]^2}\right) = \log_n(\mathrm{E}[X]) - \log_n\left(\sqrt{1 + \frac{\mathrm{Var}[X]}{\mathrm{E}[X]^2}}\right)$$
$$= \log_n\left(\frac{\mathrm{E}[X]}{\sqrt{1 + \frac{\mathrm{Var}[X]}{\mathrm{E}[X]^2}}}\right) = \log_n\left(\frac{\mathrm{E}[X]\mathrm{E}[X]}{\mathrm{E}[X]\sqrt{1 + \frac{\mathrm{Var}[X]}{\mathrm{E}[X]^2}}}\right) = \log_n\left(\frac{\mathrm{E}[X]^2}{\sqrt{\mathrm{E}[X]^2 + \mathrm{Var}[X]}}\right)$$

14.5. Parameters of the Square-Root-Normal Distribution

Given a square-root-normally distributed variable

$$X = (\mu + \sigma Z)^2 = (\mu + \sigma Z)(\mu + \sigma Z) = \mu^2 + (\sigma Z)^2 + 2\mu\sigma Z \tag{53}$$

Basic operands of the mean operator gives

$$\mathrm{E}[X] = \mathrm{E}[\mu^2 + (\sigma Z)^2 + 2\mu\sigma Z] = \mathrm{E}[\mu^2] + \mathrm{E}[\sigma^2 Z^2] + \mathrm{E}[2\mu\sigma Z] = \mu^2 + \sigma^2\mathrm{E}[Z^2] + 2\mu\sigma\mathrm{E}[Z] \tag{54}$$

given that Z follows a standard normal distribution, then $\mathrm{E}[Z] = 0$. Further it follows that $Z^2 \sim \chi^2(1)$, i.e. a chi-squared distribution with 1 degree of freedom. The mean of the χ^2 distribution is equal to the degrees of freedom and the variance is two times the degrees of freedom, consequently

$$\mathrm{E}[X] = \mu^2 + \sigma^2 \tag{55}$$

Similarly, for the variance it follows

$$\mathrm{Var}[X] = \mathrm{Var}[\mu^2 + (\sigma Z)^2 + 2\mu\sigma Z] \tag{56}$$

the term μ^2 is a constant and consequently cancels out. The terms $\sigma^2 Z^2$ and $2\mu\sigma Z^2$ are correlated (as they are both functions of Z) and consequently

$$\mathrm{Var}[X] = \sigma^4\mathrm{Var}[Z^2] + (2\mu\sigma)^2\mathrm{Var}[Z] + 4\mu\sigma^3\mathrm{Cov}[Z, Z^2] \tag{57}$$

The covariance of a distribution with its squared self is calculated as follows. Assume $X \sim \mathcal{N}(\mu, \sigma)$ then $Y = \frac{X-\mu}{\sigma} \sim \mathcal{N}(0, 1)$. Then the covariance is

$$\begin{aligned}\mathrm{Cov}[X, X^2] &= \mathrm{Cov}[\sigma Y + \mu, (\sigma Y + \mu)^2] \\ &= \mathrm{Cov}[\sigma Y + \mu, \sigma^2 Y + 2\mu\sigma Y + \mu^2] \\ &= \sigma^3\mathrm{Cov}[Y, Y^2] + 2\mu\sigma^2\mathrm{Cov}[Y, Y] \\ &= 2\mu\sigma^2\end{aligned} \tag{58}$$

The last step follows from the fact that $\mathrm{Cov}[Y, Y^2] = \mathrm{E}[Y^3] - \mathrm{E}[Y]\mathrm{E}[Y^2]$ is zero, which can be proven by use of the moment generating function of the normal distribution which shows $\mathrm{E}[Y^3] = 0$ and $\mathrm{E}[Y] = 0$. In the specific case where $Z \sim \mathcal{N}(0, 1)$, then

$$\mathrm{Cov}[Z, Z^2] = 0 \tag{59}$$

consequently it is evident that

$$\mathrm{Var}[X] = 2\sigma^4 + 4\mu^2\sigma^2 \tag{60}$$

This provides two equations with two unknowns. First isolating μ in the first equation

$$\mu = \sqrt{\mathrm{E}[X] - \sigma^2} \tag{61}$$

then substituting that into the second equation

$$\mathrm{Var}[X] = 2\sigma^4 + 4\mu^2\sigma^2 = 2\sigma^4 + 4\sigma^2(\mathrm{E}[X] - \sigma^2) = 2\sigma^4 - 4\sigma^4 + 4\sigma^2\mathrm{E}[X] = 4\sigma^2\mathrm{E}[X] - 2\sigma^4 \tag{62}$$

isolating for σ

$$\sigma = \sqrt{\mathrm{E}[X] - \sqrt{\mathrm{E}[X]^2 - \frac{\mathrm{Var}[X]}{2}}} \tag{63}$$

which is substituted back into the equation for μ

$$\mu = \sqrt{\mathrm{E}[X] - \left(\mathrm{E}[X] - \sqrt{\mathrm{E}[X]^2 - \frac{\mathrm{Var}[X]}{2}}\right)} = \sqrt[4]{\mathrm{E}[X]^2 - \frac{\mathrm{Var}[X]}{2}} \tag{64}$$

Notice that this transform is only valid for input of

$$\mathrm{E}[X] \geq \sqrt{\frac{\mathrm{Var}[X]}{2}} \tag{65}$$

14.6. Log-Uniform Independence of Logarithmic Base

Let $X = n^Z$ for $Z \sim U(\log_n(a), \log_n(b))$ where n is the logarithmic base. Then the CDF of X is

$$F(x; a, b) = \frac{\log_n(x) - \log_n(a)}{\log_n(b) - \log_n(a)} = \frac{\log_n\left(\frac{x}{a}\right)}{\log_n\left(\frac{b}{a}\right)} = \log_{\frac{b}{a}}\left(\frac{x}{a}\right) \tag{66}$$

as can be seen the last expression is completely independent of n and consequently X is independent of the choice of logarithmic base.

14.7. Log-Triangular Independence of Logarithmic Base

Let $X = n^Z$ for $Z \sim T(\log_n(a), \log_n(c), \log_n(b))$ where n is the logarithmic base. Then the CDF of X on the interval $a < x \leq c$ is

$$F(x; a, b) = \frac{\log_n\left(\frac{x}{a}\right)^2}{\log_n\left(\frac{b}{a}\right)\log_n\left(\frac{c}{a}\right)} = \frac{\log_n\left(\frac{x}{a}\right)}{\log_n\left(\frac{b}{a}\right)} \cdot \frac{\log_n\left(\frac{x}{a}\right)}{\log_n\left(\frac{c}{a}\right)} = \log_{\frac{b}{a}}\left(\frac{x}{a}\right)\log_{\frac{c}{a}}\left(\frac{x}{a}\right) \tag{67}$$

as can be seen the last expression is completely independent of n and consequently X is independent of the choice of logarithmic base. This derivation is trivial to extend to the interval $c < x \leq b$.

14.8. Log-Beta Independence of Logarithmic Base

Let $X = n^Z$ for $Z \sim B(\alpha, \beta, \log_n(a), \log_n(b))$ where n is the logarithmic base. Then the CDF of X is

$$F(x; \alpha, \beta, a, b) = F\left(\frac{\log_n(x) - \log_n(a)}{\log_n(b) - \log_n(a)}; \alpha, \beta\right) = F\left(\log_{\frac{b}{a}}\left(\frac{x}{a}\right); \alpha, \beta\right) \tag{68}$$

as can be seen the last expression is completely independent of n and consequently X is independent of the choice of logarithmic base.

15. Appendix: Geostatistical Variogram

The basis of kriging is the semivariogram and covariogram. These relate the influence of neighbouring data points to their distance from the point of estimation. In the general definition, the semivariogram is defined as

$$\gamma(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{1}{2}\text{Var}[Z(\boldsymbol{x}_1) - Z(\boldsymbol{x}_2)] = \frac{1}{2}\text{E}\left[((Z(\boldsymbol{x}_1)) - \mu(\boldsymbol{x}_1) - (Z(\boldsymbol{x}_2) - \mu(\boldsymbol{x}_2)))^2\right] \tag{69}$$

i.e. half the average squared difference between spatial point $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Further, if it can be assumed that the physical property has a constant mean over the entire field, then

$$\gamma(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{1}{2}\text{E}\left[(Z(\boldsymbol{x}_1) - Z(\boldsymbol{x}_2))^2\right] \tag{70}$$

if the process is further stationary, then

$$\gamma(\boldsymbol{x}_1, \boldsymbol{x}_2) = \gamma(\boldsymbol{x}_2 - \boldsymbol{x}_1) \tag{71}$$

and finally, if the process is isotropic, then

$$\gamma(\boldsymbol{x}_1, \boldsymbol{x}_2) = \gamma(h), \quad h = ||\boldsymbol{x}_2 - \boldsymbol{x}_1|| \tag{72}$$

i.e. the spatial dependency is only related to the distance between the spatial positions, and is independent of their location. Notice that the exact same logic applies to the covariogram. A property of the semivariogram and covariogram is that

$$C(0) = \gamma(r) = \sigma^2 \tag{73}$$

where r is the range. From this it can be shown, which can also be seen on Fig. **??**, that

$$C(h) = \sigma^2 - \gamma(h) \tag{74}$$

in theory $\gamma(0) = 0$, however due to the geographical scale on which the properties are being investigated and measurement noise, an offset from zero might be present, which is called the nugget. The sill is the maximum variability, which is reached when distance between two points is so large that spatial variability is non-existing. The range is the distance at which the sill is reached.

In reality a priori knowledge of the spatial dependency at any distance, h, from a given spatial position, is not available. To overcome this, an empirical semivariogram and covariogram is constructed instead. The distance between all samples is calculated, and binned in appropriately sized bins, then the semivariogram and covariogram may be empirically estimated using

$$\begin{aligned} \hat{\gamma}(h) &= \frac{1}{2|N(h)|}\sum_{(i,j)\in N(h)} (Z(\boldsymbol{x}_i) - Z(\boldsymbol{x}_j))^2 \\ \hat{C}(h) &= \frac{1}{|N(h)|}\sum_{(i,j)\in N(h)} (Z(\boldsymbol{x}_i) - m(h)) \cdot (Z(\boldsymbol{x}_j) - m(h)) \end{aligned} \tag{75}$$

where

$$m = \frac{1}{2|N(h)|}\sum_{(i,j)\in N(h)} Z(\boldsymbol{x}_i) + Z(\boldsymbol{x}_j) \tag{76}$$

note that $N(h) = \{(\mathbf{x}_i,\mathbf{x}_j) : ||\mathbf{x}_i - \mathbf{x}_j|| = h; i, j = 1,...,N\}$, i.e. $N(h)$, is the set of spatial samples belonging to the distance, h, plus minus some bin width. $|N(h)|$ is the frequency of spatial points belonging to the bin h.

To allow for interpolating at an arbitrary distance, h, a model is fitted to the empirical semivariogram and covariogram using a Least Squares approach. Several model exists, and a wide array of them are outlined

in Appendix A, and shown in Fig. **??** In practice so few samples may be available, that it is impractical to fit a model to an empirical semivariogram and covariogram. In such a case, an assumption of the model may be made.

16. Appendix: Optimization Benchmark Suites

16.1. Quadratic Convex Functions

The suite of quadratic convex functions are designed to test the algorithms ability to converge under ill-conditioned circumstances

Sphere

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2 \tag{77}$$

Ellipsoid

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} s^{\frac{i-1}{n-1}} x_i^2 \tag{78}$$

Cigar

$$f(\boldsymbol{x}) = x_1^2 + \sum_{i=2}^{n} s x_i^2 \tag{79}$$

Discus

$$f(\boldsymbol{x}) = s x_1^2 + \sum_{i=2}^{n} x_i^2 \tag{80}$$

Cigar-Discus

$$f(\boldsymbol{x}) = s x_1^2 + \sum_{i=2}^{n-1} s^{\frac{1}{2}} x_i^2 + x_n^2 \tag{81}$$

Two-Axes

$$f(\boldsymbol{x}) = \sum_{i=1}^{\lfloor \vartheta n \rfloor} s x_i^2 + \sum_{i=\lfloor \vartheta n \rfloor + 1}^{n} x_i^2 \tag{82}$$

The parameter s makes the problem ill-conditioned for large values, order of magnitude $s = 10^6$. The quadratic convex functions are shown for $n = 2$ on Fig. 25.

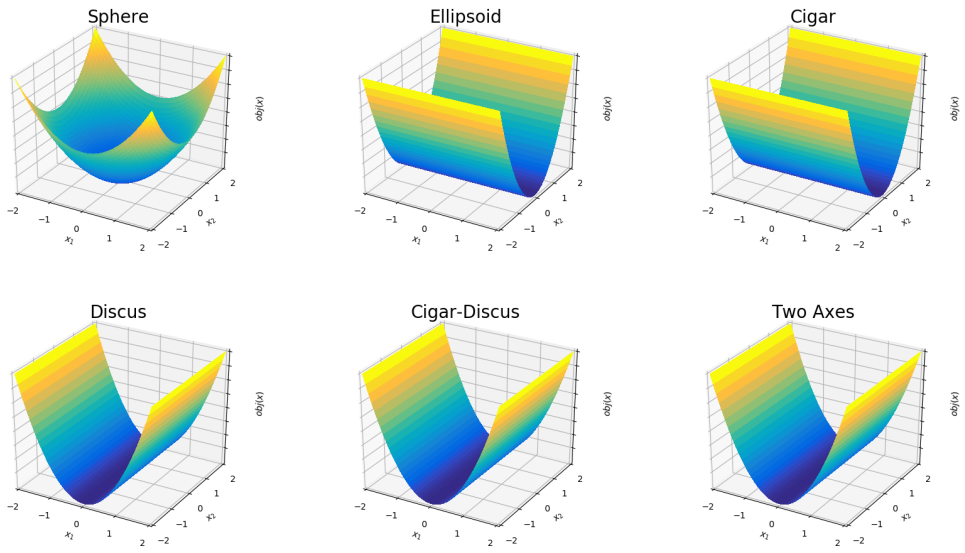


Figure 25: Quadratic convex benchmark suite with $s = 10^6$ and $\vartheta = 0.5$.

16.2. Non-Linear Functions

The suite of non-linear functions are designed to test the algorithms ability to converge to a global minimum on difficult, non-convex problems

Diff. Powers

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} |x_i|^{i+1} \tag{83}$$

Rosenbrock

$$f(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left(100 \left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right) \tag{84}$$

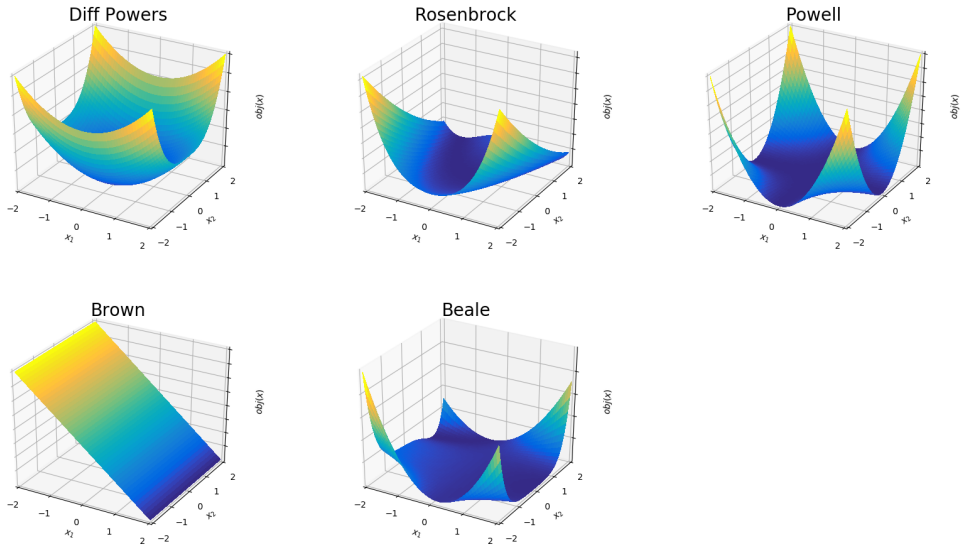


Figure 26: Non-linear benchmark suite.

16.3. Global Optimum Functions

The suite of non-linear functions are designed to test the algorithms ability to converge to a global minimum on difficult, non-convex problems

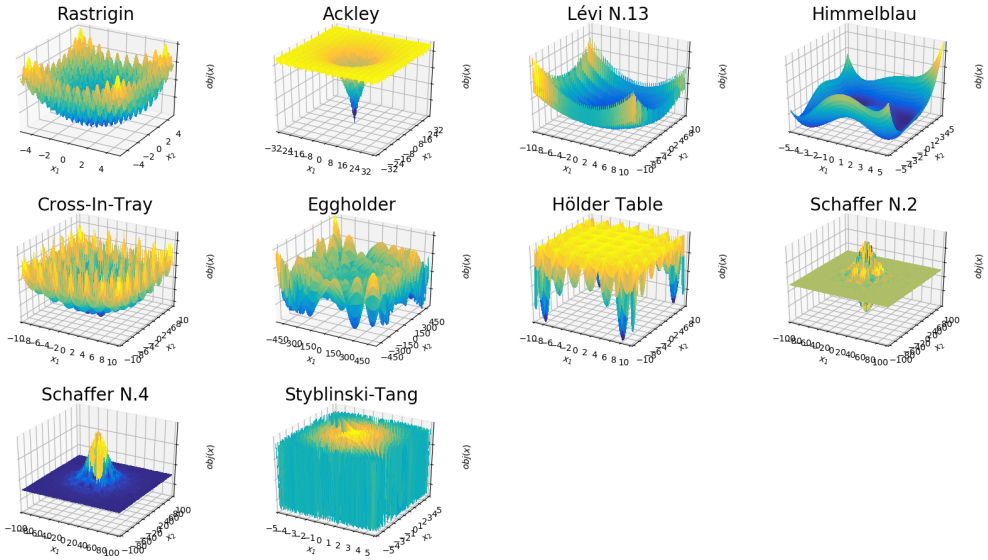


Figure 27: Global benchmark suite.

References

[1] K. Sigman (2010). *Inverse Transform Method*. Columbia University.
`http://www.columbia.edu/~ks20/4404-Sigman/4404-Notes-ITM.pdf`
Visited: 29-09-2020

[2] Phychometroscar (2019). *How to simulate correlated log-normal random variables THE RIGHT WAY*.
`https://psychometroscar.com/2019/10/16/how-to-simulate-correlated-log-normal-random-variables-the-`
Visited: 29-09-2020

[3] Mathworks. *Copulas: Generate Correlated Samples*.
`https://www.mathworks.com/help/stats/copulas-generate-correlated-samples.html`
Visited: 29-09-2020

[4] Twiecki (2018). *An intuitive, visual guide to copulas*.
`https://twiecki.io/blog/2018/05/03/copulas/`
Visited: 29-09-2020

[5] Twiecki (2015). *MCMC sampling for dummies*.
`https://twiecki.io/blog/2015/11/10/mcmc-sampling/`
Visited: 29-09-2020

[6] Robert, C. P. (2009). *Simulation of Truncated Normal Variables*. Université Pierre et Marie Curie, France.

[7] A. O'Hagan (2013). *Polynomial Chaos: A Tutorial and Critique from a Statistician's Perspective*. University of Sheffield, UK.
(see: http://tonyohagan.co.uk/academic/pdf/Polynomial-chaos.pdf)

[8] Gill, P. E. (2008). *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*. Stanford University, USA.

[9] Arnold, D., Hansen, N. (2010). *Active Covariance Matrix Adaptation for the (1+1)-CMA-ES*. HAL.

[10] Arnold, D., Hansen, N. (2012). *A (1+1)-CMA-ES for Constrained Optimisation*. HAL.

[11] Lichtenstern, A. (2013). *Kriging Methods in Spatial Statistics*. Bachelor Thesis, Technische Universität München.

[12] Lataniotis, C., Marelli, S., Sudret, B. (2018). *The Gaussian Process Modelling Module in UQLab*. ETH Zürich.

[13] Hoffman, M. D., Gelman, A. (2014). *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo*. Journal of Machine Learning Research 15 (1351-1381).

[14] Link, W. A., Eaton, M. J. (2012). *On thinning of chains in MCMC*. Methods in Ecology and Evolution 2012, 3, 112-115.

[15] Maree, S. C., Alderliesten, T., Thierens, D., Bosman, P. A. N. (2018). *Real-Valued Evolutionary Multi-Modal Optimization driven by Hill-Valley Clustering*. GECCO 18: Genetic and Evolutionary Computation Conference.