Chapter 14
The Sequential Parameter Optimization Toolbox

Thomas Bartz-Beielstein, Christian Lasarczyk, and Mike Preuss

Abstract The *sequential parameter optimization toolbox* (SPOT) is one possible implementation of the SPO framework introduced in Chap. 2. It has been successfully applied to numerous heuristics for practical and theoretical optimization problems. We describe the mechanics and interfaces employed by SPOT to enable users to plug in their own algorithms. Furthermore, two case studies are presented to demonstrate how SPOT can be applied in practice, followed by a discussion of alternative metamodels to be plugged into it. We conclude with some general guidelines.

14.1 Introduction

The sequential parameter optimization approach discussed in Chap. 2 is a flexible and general framework which can be applied in many situations. Here, we introduce the *sequential parameter optimization toolbox* (SPOT) as one possible implementation of this framework. The basic ideas of this approach can be described as follows:

1. Use the available budget sequentially, i.e., use information from the exploration of the search space to guide the search by building one or several meta models. Choose new design points based on predictions from the meta model(s). Refine the meta model(s) stepwise to improve knowledge about the search space.

Thomas Bartz-Beielstein
Institute of Computer Science, Cologne University of Applied Sciences, 51643 Gummersbach, Germany, e-mail: thomas.bartz-beielstein@fh-koeln.de

Christian Lasarczyk
Algorithm Engineering, TU Dortmund, Germany, e-mail: Christian.Lasarczyk@udo.edu

Mike Preuss
Algorithm Engineering, TU Dortmund, Germany, e-mail: mike.preuss@tu-dortmund.de

2. Try to cope with noise by improving confidence. Guarantee comparable confidence for search points.
3. Collect information to learn from this tuning process, e.g., apply explorative data analysis (Tukey 1991).
4. Provide mechanisms both for interactive and automated tuning.

We consider SPOT as one of the most effective and efficient tuning procedures that enables learning from experiments. It was developed over recent years by Thomas Bartz-Beielstein, Christian Lasarczyk, and Mike Preuss (Bartz-Beielstein et al. 2005). The main purpose of SPOT is to determine improved parameter settings for optimization algorithms to analyze and understand their performance. SPOT was successfully applied to numerous optimization algorithms, especially in the field of evolutionary computation.

The remainder of this chapter is organized as follows. Section 14.2 presents some typical examples. Section 14.3 describes goals of the SPOT approach. In Sect. 14.4, elements of SPOT are explained. Since SPOT requires some statistical considerations, these will be discussed in Sect. 14.5. A case study that performs a regression analysis of the (1+1)-ES from Chap. 2 is presented in Sect. 14.6. Besides classical regression, SPOT allows the use of modern models such as tree-based regressions; pros and cons of these models are discussed in Sect. 14.7. The sequential approach is reconsidered in Sect. 14.8. Results from this chapter are summarized in Sect. 14.9.

14.2 Applications

This section presents typical applications of the sequential parameter optimization toolbox from bioinformatics, water-resource management, mechanical engineering, biogas plant simulation, shipbuilding, and quality control. Bartz-Beielstein (2010a) lists more than one hundred publications which mention how SPOT can be applied to evolution strategies, particle swarm optimization, genetic programming, and many more algorithms.

14.2.1 Bioinformatics

Volkert (2006) discusses results from a sequential parameter optimization approach for investigating the effectiveness of using EAs for training *hidden Markov models* (HMM). She concludes that "this approach not only offers the possibility for improving the effectiveness of the EA but will also provide much needed insight into directions for future improvements in the design of EAs for the construction of HMMs in general."

Fober et al. (2009) adjusted seven exogenous parameters of an EA using the sequential parameter optimization toolbox. The EA was used for a structural analysis of biomolecules.

14.2.2 Water-Resource Management

The aim of a paper by Konen et al. (2009) is the prediction of fill levels in stormwater tanks based on rain measurements and soil conditions. Different prediction methods are compared. The sequential parameter optimization is used to find in a comparable manner the best parameters for each method. Main results of this work are: (i) SPOT is applicable to diverse forecasting methods and automates the time-consuming parameter tuning, (ii) the best manual result achieved before was improved with SPOT by 30% and (iii) SPOT analyses in a consistent manner the parameter influence and allows a purposeful simplification and/or refinement of the model design.

14.2.3 Mechanical Engineering

Mehnen et al. (2007) apply SPOT in mechanical engineering for the design of *mold temperature control strategies* (MTCS), which is a challenging multi-objective optimization task. In this paper an EA is applied to a multi-objective problem using aggregation. The DACE technique is used to find good *multi-objective evolutionary algorithm* (MOEA) parameter settings to get improved solutions of the MTCS problem. An automatic and integrated software package, which is based on the DACE approach, is applied to find the statistically significant and most promising EA parameters using SPOT.

14.2.4 Biogas

Ziegenhirt et al. (2010) use SPOT to optimize the simulation of biogas plants. There is a high demand from industry to run these plants efficiently, which leads to a highly complex simulation and optimization problem. A comparison of several algorithms from computational intelligence to solve this problem is presented in their study. The sequential parameter optimization was used to determine improved parameter settings for these algorithms in an automated manner. They demonstrate that genetic algorithm and particle swarm optimization based approaches were outperformed by differential evolution and covariance matrix adaptation evolution strategy. Compared to previously presented results, their approach required only one tenth of the number of function evaluations.

14.2.5 Shipbuilding

Rudolph et al. (2009) discuss the optimization of a relatively new ship propulsion system (a linear jet) which possesses 15 design variables. It consists of a tube with a rotor and a stator, and several lengths, angles, and thicknesses that can be variated. The objective function is a very basic fluid dynamic simulation of a linear jet that takes about 3 minutes to compute, and the task is to reduce cavitation at a predefined efficiency. A full simulation would take about 8 hours, which is by far too much to serve as test problem. A surrogate model is used to detect good design points that can afterwards be validated by the full simulation. The authors conclude: "The validation experiment is successful, as the SPO-tuned parameter values lead to a significant performance increase. The tuned MAES resembles a model-enhanced (4,11)-ES with $\kappa = 20$ and so may profit from a more globally oriented behavior than the standard parameter setting with a population size of one."

14.2.6 Fuzzy Operator Trees

Yi (2008) proposes a method for modeling utility (rating) functions based on a novel concept called *fuzzy operator tree* (FOT). As the notion suggests, this method makes use of techniques from fuzzy set theory and implements a fuzzy rating function, that is, a utility function that maps to the unit interval, where 0 corresponds to the lowest and 1 to the highest evaluation. Even though the original motivation comes from quality control, FOTs are completely general and widely applicable. Yi (2008) reports that "several works have shown that SPO outperforms other alternatives, especially for evolution strategies, we shall apply SPO to determine the parameters of ES in this section. Sequential sampling approaches with adaptation have been proposed for DACE, here let us review the basic idea of Thomas Bartz-Beielstein, which is also used in this section to determine the parameters of ES."

14.3 Objectives

During the first stage of experimentation, SPOT treats the algorithm A as a black box. A set of input variables, say $\mathbf{x}$, is passed to A. Each run of the algorithm produces some output, $\mathbf{y}$. SPOT tries to determine a functional relationship F between $\mathbf{x}$ and $\mathbf{y}$ for a given problem formulated by an objective function $f : \mathbf{u} \to \mathbf{v}$. Since experiments are run on computers, pseudorandom numbers are taken into consideration if:

(i) The underlying objective function f is stochastically disturbed, e.g., measurement errors or noise occur, and/or

(ii) The algorithm A uses some stochastic elements, e.g., mutation in evolution strategies.

This situation can be described as follows:

$$\text{Objective function:} \quad \mathbf{u} \xrightarrow{f} \mathbf{v}, \tag{14.1}$$

$$\text{Algorithm:} \quad \mathbf{x} \xrightarrow{F} \mathbf{y}. \tag{14.2}$$

We will classify elements from (14.1) and (14.2) in the following manner, see also Sect. 2.2.1:

1. Variables that are necessary for the algorithm belong to the algorithm design, whereas
2. variables that are needed to specify the optimization problem f belong to the problem design.

SPOT employs a sequentially improved model to estimate the relationship between algorithm input variables and its output. This serves two primary goals. One is to enable determining good parameter settings, thus SPOT may be used as a tuner. Secondly, variable interactions can be revealed that help to understand how the tested algorithm works when confronted with a specific problem or how changes in the problem influence the algorithm's performance. Concerning the model, SPOT allows for insertion of virtually every available model. However, regression and Kriging models or a combination thereof are most frequently used.

14.4 Elements of the SPOT Framework

In this section, we describe SPOT as one possible implementation of steps 3a)-d) from the general SPO scheme (see Sect. 2.5.5.4) and discuss the different tools of the framework that may be employed separately or in automated mode. We conclude by giving a definition and a short introduction to starting and using SPOT.

14.4.1 The General SPOT Scheme

Definition 14.1 (Sequential Parameter Optimization Toolbox). The sequential parameter optimization toolbox implements the following features:

SPOT-1: Use the available budget (e.g., simulator runs, number of function evaluations) sequentially, i.e., use information from the exploration of the search space to guide the search by building one or several meta models. Choose new design points based on predictions from the meta model(s). Refine the meta model(s)) stepwise to improve knowledge about the search space.

SPOT-2: Try to cope with noise by improving confidence. Guarantee comparable confidence for search points.

SPOT-3: Collect information to learn from this tuning process, e.g., apply explorative data analysis.
SPOT-4: Provide mechanisms both for interactive and automated tuning.

□

Algorithm 14.1 presents a formal description of the SPOT scheme. This scheme consists of two phases, namely the first construction of the model and its sequential improvement. Phase 1 determines a population of initial designs in algorithm parameter space and runs the algorithm k times for each design. Phase 2 consists of a loop with the following components: By means of the obtained data, the model is built or updated, respectively. Then, a possibly large set of design points is generated and their predicted utility computed by sampling the model. A small set of the seemingly best design points is selected and the algorithm is run $k + 1$ times for each of these. The algorithm is also run once for the current best design point and k is increased by one. Note, other update rules for the number of repeats, k, are possible. The new design points are added to the population and the loop starts over if the termination criterion is not reached (usually a preset budget is granted to the process). In consequence, this means that the number of repeats is always increased by one if the current best design point stays at the top of the list or a newly generated one gets there. Due to nondeterministic responses of the algorithm, it may however happen that neither of these is found at the top of the list after finishing the loop. In this case, k may effectively shrink as performance comparisons have to be fair and thus shall be based on the same number of repeats.

Sequential approaches are generally more efficient, i.e., require fewer function evaluations, than approaches that evaluate the information in one step only. Chapter 15 of this book discusses an extension of this sequential framework. Further extensions were proposed by other authors, e.g., Lasarczyk (2007) integrated an *optimal computational budget allocation* (OCBA) procedure, which is based on ideas by Chen et al. (2003).

14.4.2 SPOT Tasks

If used for tuning an algorithm, SPOT is not per se envisioned as a meta-algorithm. We are interested in the resulting algorithm designs, not in the solutions to the primordial problem. However, it is of course possible to model a problem directly, without employing an algorithm. SPOT provides tools to perform the following tasks:

1. *Initialize*. An initial design is generated and written to a design file. This is usually the first step during experimentation. The employed parameter region and the constant algorithm parameters have to be provided by the user. A clean start can be performed. This is necessary if the user wants to run a new experiment or delete results from previous steps.

Algorithm 14.1: Sequential parameter optimization toolbox (SPOT)

```
// phase 1, building the model:
let $A$ be the tuned algorithm;
generate an initial population $X = \{\bar{x}^1, \dots, \bar{x}^m\}$ of $m$ parameter vectors;
let $k = k_0$ be the initial number of tests for determining estimated utilities;
foreach $\bar{x} \in X$ do
    run $A$ with $\bar{x}$ $k$ times to determine the estimated utility $y$ of $\bar{x}$;
end
// phase 2, using and improving the model:
while termination criterion not true do
    let $\bar{a}$ denote the parameter vector from $X$ with best estimated utility;
    let $k$ the number of repeats already computed for $\bar{a}$;
    build prediction model $f$ based on $X$ and $\{y^1, \dots, y^{|X|}\}$;
    generate a set $X'$ of $l$ new parameter vectors by random sampling;
    foreach $\bar{x} \in X'$ do
        calculate $f(\bar{x})$ to determine the predicted utility $f(\bar{x})$ of $\bar{x}$;
    end
    select set $X''$ of $d$ parameter vectors from $X'$ with best predicted utility ($d \ll l$);
    run $A$ with $\bar{a}$ once and recalculate its estimated utility using all $k + 1$ test results;
      // (improve confidence)
    let $k = k + 1$;
    run $A$ $k$ times with each $\bar{x} \in X''$ to determine the estimated utility $\bar{x}$;
    extend the population by $X = X \cup X''$;
end
```

2. *Run.* This is usually the second step. The optimization algorithm is started with configurations from the design file. The algorithm writes results to the result file. Information from the design and algorithm problem design files are used in this step.
3. *Sequential step.* A new design, based on information from the result file, is generated. This new design is written to the design file. A prediction model is used in this step. Several generic prediction models are available in SPOT already. To perform an efficient analysis, especially in situations when only few algorithms runs are possible, user-specified prediction models can easily be integrated into SPOT.
4. *Report.* An analysis, based on information from the result file, is generated. Since the report uses information from the result file, new report facilities can be added very easily. SPOT contains some scripts to perform a basic regression analysis and plots such as histograms, scatter plots, plots of the residuals, etc.
5. *Automatic* mode. In the automatic mode, the steps *run* and *sequential* are performed after the initialization a certain number of times.
6. *Meta* mode. In the meta mode, several problem instances can be used for tuning.

14.4.3 Running SPOT

Previous versions of the SPOT relied on functions provided by the MATLAB Kriging toolbox DACE developed by Lophaven et al. (2002). Starting with version 0.5, an R (Ihaka and Gentleman 1996) version is available, too. The following description refers to this R implementation (Bartz-Beielstein 2010b). Of course, an R-system must be available on your computer. The SPOT package can be obtained via CRAN, the *Comprehensive R Archive Network*, see `http://cran.r-project.org`. Therefore, the simplest way to install the package is to enter

```
install.packages("SPOT")
```

into your R session. SPOT can be loaded via

```
library("SPOT")
```

The formal command to start SPOT tasks reads:

```
spot( <configurationfile>, <task>)
```

where `task` can be one of the tasks described in Sect. 14.4.2, i.e., init, seq, run, rep, auto, or meta, and `configurationfile` is the name of the SPOT configuration

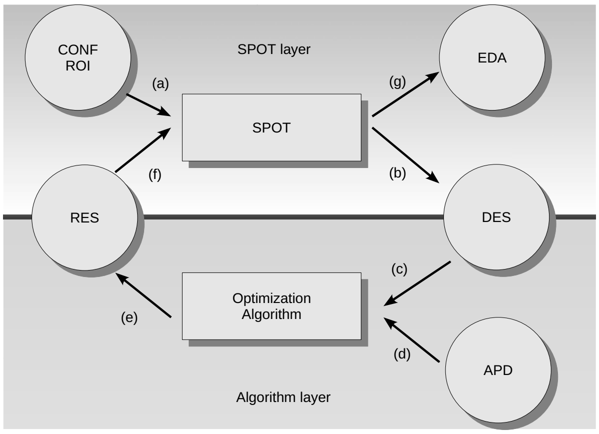

Fig. 14.1: SPOT interfaces. The SPOT loop can be described as follows: *Configuration* (CONF) and *region-of-interest* (ROI) files are read by SPOT (a). SPOT generates a *design* (DES) file (b). The algorithm reads the design file and (c) extra information, e.g., about the problem dimension from the *algorithm-problem design* (APD) file (d). Output from the optimization algorithm are written to the *result* (RES) file (e). The result file is used by SPOT to build the prediction model (f). Data can be used by *exploratory data analysis* (EDA) tools to generate reports, statistics, visualizations, etc. (g)

file. SPOT uses simple text files as interfaces to the algorithm to the statistical tools. A JAVA-based GUI, which simplifies parameter input, is available, too.

1. Files provided by the user:
   (a) *Region of interest* (ROI) files specify the region over which the algorithm parameters are tuned. Categorical variables such as the recombination operator in ES, are encoded as numerical values, e.g., "1" represents "no recombination" and "2" stands for "discrete recombination."
   (b) *Algorithm design* (APD) files are used to specify parameters used by the algorithm, e.g., problem dimension, objective function, starting point or initial seed.
   (c) *Configuration* files (CONF) specify SPOT specific parameters, such as the prediction model or the initial design size.
2. Files generated by SPOT:
   (a) *Design* files (DES) specify algorithm designs. They are generated automatically by SPOT and will be read by the optimization algorithms.
   (b) After the algorithm has been started with a parametrization from the algorithm design, the algorithm writes its results to the *result file* (RES). Result files provide the basis for many statistical evaluations/visualizations. They are read by SPOT to generate prediction models. Additional prediction models can easily be integrated into SPOT.

Figure 14.1 illustrates SPOT interfaces and the data flow. Note, that the problem design can be modified too. This can be done to analyze the robustness (effectivity) of algorithms.

SPOT can be run in an *automated* and an *interactive mode*. Similarities and differences of the automated and the interactive process are shown in Fig. 14.2.

14.5 Statistical Considerations

14.5.1 Sequential Models

To introduce SPOT's functionality, we describe a case study that analyzes the evolution strategy introduced in Chap. 2. Starting from scratch, we do not know anything about the functional relationship F from (14.1). This situation can be characterized as a chicken-and-egg problem: The experimenter has to decide which comes first: a design for the data $\mathbf{x}$ or the specification of a model, i.e., by defining a functional relationship F? For example, assuming a linear relationship F, we should use factorial designs to specify $\mathbf{x}$. However, in order to validate linearity, we need some data $\mathbf{x}$.

We prefer the following approach: Select a simple model (functional relationship) F_0 and a related design $\mathbf{x}_0$, generate some data $\mathbf{y}_0$, fit the (simple) model F_0,

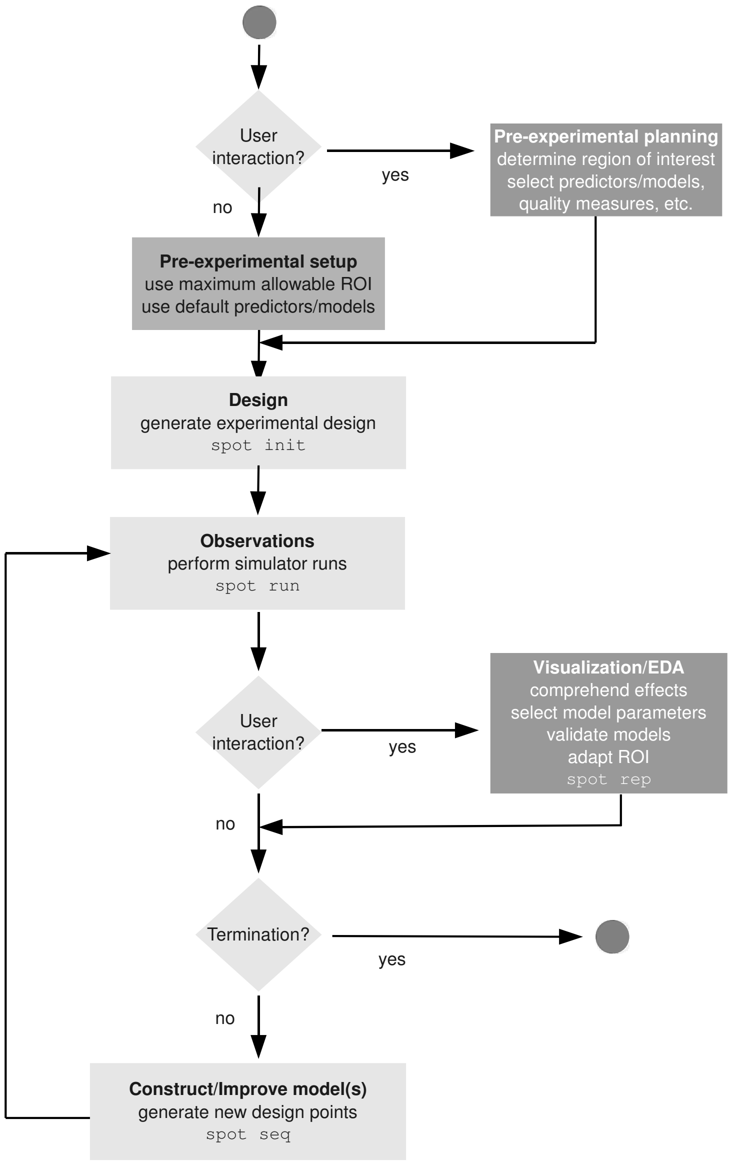

Fig. 14.2: The sequential parameter optimization process. *White font color* indicates elements that are used in the interactive process only. A *typewriter font* indicates the corresponding SPOT commands. To start the automated mode, simply use the task `auto`. Note that the interaction points are optional, so SPOT can be run without any user interaction

predict a few new design points $\mathbf{x}_1$ based on F_0, generate further data $\mathbf{y}_1$, refine the model (F_1), and continue. The chicken-and-egg problem defines a sequential approach in a natural manner, which improves F and y at the same time. This approach motivated the term *sequential parameter optimization*. In many situations, it can be shown that sequential approaches are much more efficient than one-at-a-time approaches (Armitage 1975).

We will discuss the basic output from the R analysis during the sequential approach. The analysis is not complicated and requires only basic knowledge of statistics. The reader will get interesting insights into the working mechanism of his algorithms by executing a few lines of R code. SPOT comes with some elementary R scripts which can be easily extended.

First, we will focus on fitting a linear regression model. The model

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \epsilon \tag{14.3}$$

is called a multiple linear regression model with k *regression variables* and *response* y. The regression variables represent algorithm parameters, e.g., initial step size s_0, multiplier for step sizes a, and size of the memory g. A variable that can be used to predict the value of another variable is also called an "input variable," "prediction variable," or "regressor." Response variables are also called "output variables." The corresponding predictive equation reads

$$\hat{\mathbf{y}} = \mathbf{b}\mathbf{X}. \tag{14.4}$$

The small roman letter $\mathbf{b}$ denotes a vector of the estimates of the parameters $\beta_0, \ldots, \beta_k$.

R uses *Wilkinson–Rogers notation* to describe models, e.g.,

```
y ~ x1,
```

where y denotes the response, "$\sim$" can be read as "is modeled by," and $x1$ is the explanatory variable. Two or more explanatory variables can be incorporated into the model as shown in

```
y ~ x1 + x2.
```

There are more advanced ways of generating models, e.g., `A:B` for interactions, where `A:B` is a shorthand notation for `A+B+A*B`, etc.

Equation (14.3) describes a hyperplane in the k-dimensional space of the regressor variables x_j ($j = 1, \ldots, k$). Assuming that all remaining independent variables x_i were held constant, the parameter β_j describes the expected change in y (response) per unit change in x_j ($i \neq j$).[1] We will discuss some considerations which are useful in the SPOT framework. The approach presented in this chapter relies on standard regression techniques only. Other models can be integrated into SPOT as well, e.g., tree-based regression or Kriging.

[1] A comprehensive introduction to regression is given in the appendix of this book.

Until now, no statistical assumptions that involve probability distributions have been made at all. In many situations, the following assumptions, which can be used to examine the regression coefficients, are made:

(a) the residual ϵ_i is a random variable with mean zero and unknown variance σ^2, i.e., $E(\epsilon_i) = 0$ and $V(\epsilon_i) = \sigma^2$,
(b) the ϵ's are uncorrelated, i.e., $\mathrm{Cov}(\epsilon_i, \epsilon_j) = 0$.
(c) $\epsilon_i \sim N(0, \sigma^2)$, i.e., ϵ_i is a normally distributed random variable, with mean zero and variance σ^2.

Assuming that the errors ϵ_i are all from the same normal distribution, the following test can be performed: The null hypothesis, i.e., β_1 equals β_{10}, especially $\beta_{10} = 0$, can be tested against the alternative that β_1 is different from β_{10} by calculating $t = \frac{b_1 - \beta_{10}}{\mathrm{se}(b_1)}$ (where $\mathrm{se}(b_1)$ denotes the standard error of the estimate of the parameter β_1) and comparing $|t|$ with $t(n-2, 1-\alpha/2)$, where $t(n, 1-\alpha)$ is the $100(1-\alpha)$ percentage point of a t-distribution with n degrees of freedom.

14.5.2 Residuals and Variance

The considerations in this section are not restricted to linear models only; they apply to any situation when a model is fitted. *Residuals* are, per definition, differences between values from actual observations and predictions by the regression equation. They describe the amount that the regression equation has not been able to explain. The residuals can be interpreted as the observed errors if the model is correct. We discuss graphical methods to examine residuals, especially plots of residuals versus fitted values, which could indicate that there should be a curve rather than a line, and normal probability plots of the residuals.

The mean square about the regression, s^2, will provide an estimate of the variance about the regression. If the regression equation (14.3) were estimated from an infinitely large number of experiments, the variance about the regression would provide a measure of the error with which any observed value of Y could be predicted from a given value of X using (14.3).

14.5.3 Standardized Variables and Transformations

The magnitude of the regression coefficients can be used to measure the effect of a variable. Problems can occur, because the regression coefficients depend on the underlying scale of measurements. For example, the coefficient for `time` measures the expected difference in response for each hour of difference in `time`. If `time` were measured in seconds instead of hours, the regression coefficient would be multiplied by 3,600, although the change in units does not change a variable's importance. To solve the problem of units of measurement, standardized regression coefficients can be used. We consider regression models with parameters β and standardized variables, i.e., each x_i ranges between -1 and $+1$. Although standardization appears appealing at the first sight, its usage is controversial; see, e.g., the discussion in Kleijnen (1987).

We have applied transformations on the function values, especially $\log$-transformations, to improve the model fit. In general, we recommend using the raw data, e.g., to improve the interpretability of the results. In some situations, lack of fit was determined, to determine the most adequate model (linear versus quadratic etc.).

14.5.4 Design Considerations and the Region of Interest

Several designs can be used to generate initial data. Principally, two design types can be distinguished:

- Factorial designs, which are commonly used in classical regression to fit linear models, place design points at the border of the region of interest (Kleijnen 1987).
- Modern approaches such as Kriging use space-filling designs, e.g., Latin hypercube designs, which place design points in the interior of the region of interest (Santner et al. 2003).

The selection of an optimal design depends on the model which will be used for prediction. The pre-experimental planning phase includes test runs in a situation where no information about the model is available and no optimality conditions from DOE, e.g., as described in Pukelsheim (1993), are applicable. The main goal of this phase is to detect intervals for experimentation, i.e., the *region of interest* (ROI). As a rule of thumb, we can state that these intervals should be chosen courageously, because SPOT should be able to guide the search into promising regions of the search space. For some settings, the experimenter has to set up rules for the treatment of infeasible factor settings, e.g., by defining a penalty function or by introducing rounding mechanisms to ensure integer values.

Draper and Smith (1998, p. 86) discuss practical design-of-experiment implications of regression. They describe the influence of experimental arrangements for obtaining data for fitting a simple linear model. The question reads: At what values (sites) should how many experimental runs be performed? The relationship between lack of fit, pure error, $\mathrm{sd}(b_1)/\sigma$, and the number of sites is presented. They consider a situation in which n experiments can be performed and σ^2 is unknown. Performing n experiments at n different sites is a poor choice, because σ^2 cannot be estimated. On the other hand, performing $\lfloor n/2 \rfloor$ repeats at only two design sites results in no degrees of freedom left for estimating the lack of fit. On the basis of the $\mathrm{sd}(b_1)/\sigma$ values, configurations with many repeats are preferable. The best choice lies in arrangements that allow testing for lack of fit and minimization of the $\mathrm{sd}(b_1)/\sigma$ values.

Example 14.1. Suppose 14 runs are possible to fit a straight line. The range is the interval $[-1, +1]$ of the coded predictor. Variance σ^2 is unknown. Choosing 14 de-

Table 14.1: Regions of interest used in the pre-experimental planning phase. S0 and A denote real variables, whereas G is an integer value

Factor	Low	High	Type
S0	0.1	5	FLOAT
A	0	2	FLOAT
G	1	100	INT

sign points with only one repeat at each point is a poor choice, because σ^2 cannot be estimated. However, also the use of seven repeats at two design points is not recommended, because lack of fit (to test the adequacy of a linear model) cannot be checked (Draper and Smith 1998). □

14.6 A Case Study: Simple Regression Applied to Evolution Strategies

Classical *response surface methods* (RSM) use three steps: screening, modeling, and optimization (Kleijnen 1987, Montgomery 2001). Design complexity is increased during the RSM process, so complex models can be fitted in the last phase of the RSM process. SPOT is closely related to RSM. Because SPOT includes rules to analyze the scientific relevance (severity) of results from the statistical analysis and specifies rules for learning from error, it can be seen as an extension of the classical RSM framework.

14.6.1 Pre-experimental Planning

This section describes how basic ideas from regression analysis can be applied to analyze evolution strategies. SPOT allows the specification of:

a) upper and lower limits of the region of interest (experimental region),
b) the number of repeats for the initial design, and
c) the type of design to be generated, e.g., LHD.

We have chosen a Latin hypercube design for the first experiments, see Table 14.1. This design consists of three factors with different types. Although factorial designs are recommended for DOE (see also the discussion in Sect. 14.5.4), a space-filling design was chosen for the pre-experimental planning phase. This design was chosen because we expect interesting model features in the center of the experimental region, and we are interested in using other models, e.g., tree-based regression models, or Kriging, in parallel. These models require space-filling designs. So, choosing

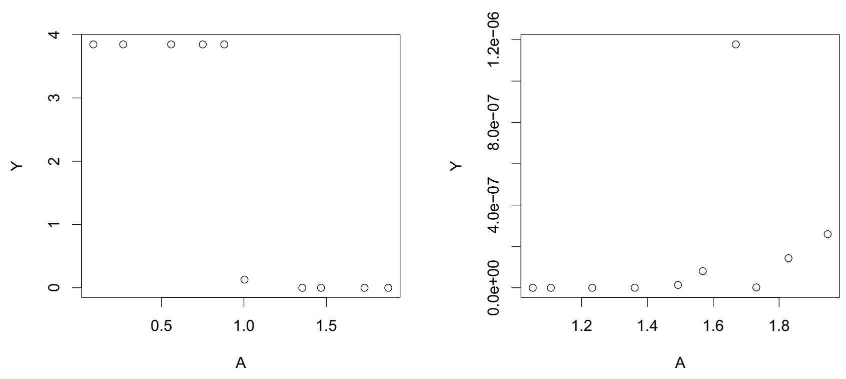

Fig. 14.3: Scatter plots from the pre-experimental planning phase to determine the region of interest. *Left:* A range from 0 to 2 for the multiplier a was chosen. Values smaller than 1 invert the 1/5th rule (2.4). As can be seen from the figures, values smaller than 1 worsen the performance of the (1+1)-ES. *Right:* If a is chosen from the interval between 1 and 2, the (1+1)-ES performs quite reasonably. These results support the recommendations given in (2.4)

an LHD can be seen as a good compromise, especially if we do not know whether linear regression models are adequate.

A *scatter* plot was used to analyze results from the first design. It is based on ten design points, where each design point represents one algorithm configuration. Each run was performed once. Figure 14.3 clearly illustrates that the multiplier for the step length should be larger than 1.0. This supports the assumption that step sizes should be increased if the success rate is larger than $1/5$. Values smaller than 1.0 are excluded from the following investigations. By repeating the same configuration as in the first study, but with values for A chosen from the interval $[1, 2]$, significant improvements can be observed. The algorithm performs quite well with parameters chosen from these ROI. Therefore, we decided to start the analysis with a multiplier of the step width chosen from the interval $[1, 2]$.

14.6.2 Performing the First Regression Analysis

To regress fitness, y, on the these quantitative predictors, we obtain the output in R as shown in Fig. 14.4.

The R output gives a numerical summary of the residuals and a table of the regression parameters. For example, the coefficient b_1 is the change in the response y when x_1 is changed by one unit. Besides estimated regression coefficients and their standard errors, t-statistics and p-values for the individual tests of the hypotheses are included. Here, we see that the intercept of the fitted line is $b_0 = -28.0456$

```
Call:
lm(formula = log(Y) ~ log(S0) + log(A) + log(G), data = df0002)
Residuals:
    Min      1Q  Median      3Q     Max
-3.6093 -1.7175  0.7059  1.4426  3.0768
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -28.0456     5.6820  -4.936  0.00261 **
log(S0)       0.4115     0.9836   0.418  0.69022
log(A)       27.2698     4.8056   5.675  0.00129 **
log(G)       -0.9124     1.2219  -0.747  0.48346
---
Residual standard error: 2.791 on 6 degrees of freedom
Multiple R-squared: 0.8671,Adjusted R-squared: 0.8006
F-statistic: 13.05 on 3 and 6 DF,  p-value: 0.004872
```

Fig. 14.4: Output from the first regression model in R.

with $\text{se}(b_0) = 5.6820$, and the estimated regression coefficient b_1 is 0.4115 with $\text{se}(b_1) = 0.9836$. The multiplier A has the largest effect, because its value reads 27.2698. The memory G has only minor impact on the performance of the ES. The table reports also t-statistics and p-values for individual tests of the hypotheses that the true intercept is zero and that the true slope is zero. The `Residual standard error` is an expression of the variation of the observations around the fitted line. It can be used to estimate σ. `Multiple R-Squared` and `Adjusted R-Squared` are reported as well. The values related to the `F-statistic` describe an F-test for the hypothesis that the regression coefficients are zero.

In addition, an *analysis of variance* (ANOVA) table can be generated. Based on an ANOVA table we can compare two models:

(a) Model M_1, which includes $S0$, A, and G
(b) Model M_2, which includes the parameter with the highest significance, namely A

As expected, this indicates that there is no significant improvement of the model once $S0$ and G are included. Removing the parameters $S0$ and G from the model is recommended. R has a built-in function which adds parameters one at a time to the current model. The `add1` function adds parameters one after another from a list and shows the resulting statistics. The default output table reports the *Akaike information criterion* (AIC), defined as minus twice the log likelihood plus $2p$, where p is the rank of the model (the number of effective parameters). For performing model searches by AIC, R provides the `stepAIC` function. Since selection and adjustment of information criteria is a difficult task (and beyond the scope of this introduction), we simply show the output from `stepAIC` applied to our example in Fig. 14.5.

The final model, suggested by `stepAIC`, includes the parameter A only.

Residual plots can be used to check model assumptions. Common checks include:

```
Call:
lm(formula = Y ~ A, data = df0002normlogy)
Residuals:
    Min      1Q  Median      3Q     Max
-3.7071 -2.6383  0.2352  2.4411  3.8783
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -20.6070     0.9672 -21.305 2.48e-08 ***
A             8.1253     1.4940   5.439 0.000617 ***
---
Residual standard error: 3.059 on 8 degrees of freedom
Multiple R-squared: 0.7871,Adjusted R-squared: 0.7605
F-statistic: 29.58 on 1 and 8 DF,  p-value: 0.000617
```

Fig. 14.5: Output from the model based on R's `stepAIC` function

a) A plot of residuals versus fitted values
b) Normal probability plots of residuals

In addition, visual inspections, e.g., on the basis of *added-variable plots* (Fig. 14.6) can be performed. Added-variable plots focus on one variable at a time and take into account the influence of the other predictors (Draper and Smith 1998, Fox 2002). To determine the influence of predictor x_j on the response y, added-variable plots are generated as follows. Let x_{-i} denote the set of regressors

$$\{x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}.$$

1. Regress y on x_{-j} , obtaining residuals $e(j)$.
2. Regress x_j on predictors x_{-j} , obtaining residuals $x(j)$.
3. Plot $e(j)$ versus $x(j)$.

All ES runs were performed using the same number of function evaluations. The results clearly indicate that A should be reduced. Smaller values improve the algorithms performance. The linear model was able to detect this trend. This was the first step of the analysis. The experimental setup shown in Table 14.1 includes a systematic variation of three parameters.

Summarizing Results from the Regression Model Analysis

The regression models used in this study illustrate that A should be decreased. Stepwise regression results in the simplified model

$$\hat{y} = -20.61 + 8.12a,$$

which indicates that the multiplier for the step sizes should be decreased. A similar result was obtained from a tree-based regression. The result occurs independently from the chosen model.

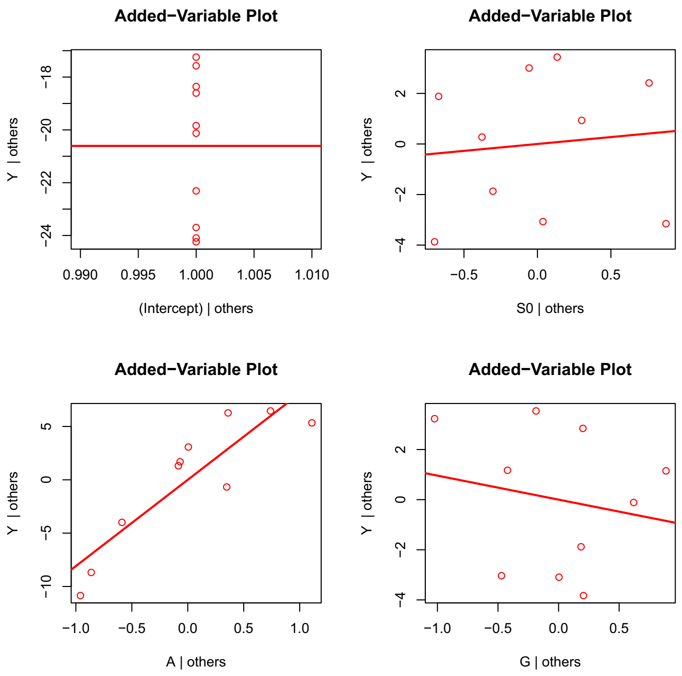

Fig. 14.6: Added-variable plot based on ten runs from the initial design of ES on the ten-dimensional sphere function. Smaller y values represent better solutions

How should one proceed if interactions that play an important role in the regression model might be of interest? We recommend *not* to apply the method of steepest descent based on the main factor model if there are significant interactions in the model. Instead, the optimum predicted by the reduced regression model including the main effects and the interactions can be used (Mehnen et al. 2004).

14.6.3 Steepest Descent

Results from the regression analysis provide information about the steepest descent in a natural manner. We proceed with the steepest descent based on the model `Y ~ S0 + A + G`. The procedure of *steepest descent* is performed as follows: Starting from the design center, we move sequentially in the direction of the maximum decrease in the response. This direction is parallel to the normal of the fitted response surface. Usually in RSM, the path of the steepest descent is taken as the line through the center of the ROI and normal to the fitted surface, i.e., the steps are proportional to the β_i's (regression coefficients). In general, the following procedure

Table 14.2: Steepest descent. $S0$ and A denote real variables, whereas G is an integer value, so its values are rounded to the next integer

	S0	A	G
1	2.54	1.50	52.50
2	2.52	1.45	53.05
3	2.50	1.41	53.60
4	2.49	1.37	54.15
5	2.47	1.32	54.70
6	2.45	1.28	55.25
7	2.44	1.23	55.80
8	2.42	1.19	56.35
9	2.40	1.14	56.90
10	2.39	1.10	57.45
11	2.37	1.05	58.01

for determining the coordinates of the points on the path of the steepest descent can be applied (Montgomery 2001):

1. Select the variable we know most about or the variable that has the largest absolute regression coefficient $|b_i|$.
2. Determine the step size in the other variables as

$$\Delta x_i = \frac{b_i}{b_j} \Delta x_j \qquad i = 1, 2, \ldots, k; \qquad i \neq j.$$

3. Convert the Δx_i stepwidths from the coded to the δx_i in the natural variables.

As the largest step size we recommend the value that leads to the border of the ROI. Data from the steepest descent experiments are shown in Table 14.2. Note that the steepest-descent experiments do not have to be performed in sequence. For example, run 10 can be performed before run 2, or a simple interval search can be performed. We recommend generating a graph of these results to determine the new region of interest following the direction of the steepest descent, cf. Fig. 14.7. The region of interest is moved down the response surface.

These settings are used for additional runs of ES. Figure 14.7 plots the yield at each step along the path of the steepest descent.

Based on visual inspection of the yields in Fig. 14.7, the new central point was determined to be the tenth point of the steepest descent. This new central point leaves some space for variation of the A values, say in the interval $[1.01, 1.2]$. Based on the best value obtained with the steepest descent, we build a new model with center point

$$\mathbf{x}_c = [S0, A, G] = [2.5, 1.125, 40].$$

The specification of the new region of interest requires user knowledge. The new center point was determined by interpreting graphical results which were based on the steepest descent. Next, we have to determine a new region around $\mathbf{x}_c$. Sometimes, especially when a classical factorial design is used during the first step, it can

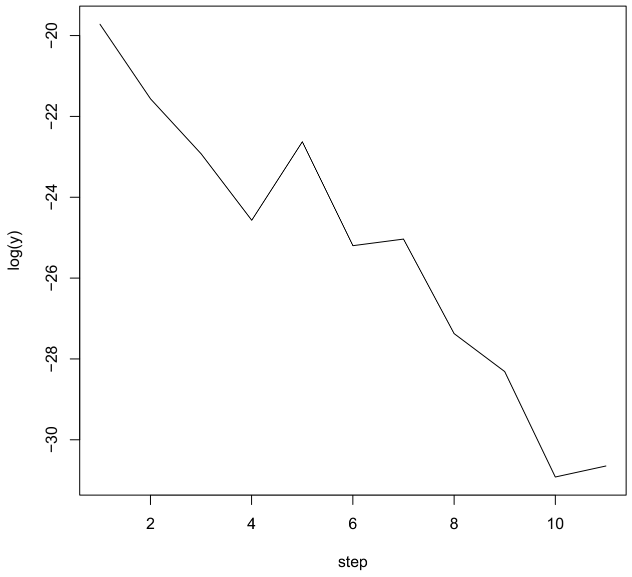

Fig. 14.7: Function values $f(x)$ (sphere) versus steps along the path of the steepest descent. Indices denote the ten steps from the steepest descent. Note, that these values are based on one repeat only, so variation in the data, e.g., the peak (index 5), is not surprising

be useful to increase the region of interest at this stage. However, we have chosen a space-filling design for the first step. Therefore, we have to decrease the region of interest. As a rule of thumb, which is to be reconsidered in any situation, we use at least $\pm\ 1/5$th of the values at the new central point. For example, if the value of the new initial step size $S0$ is 5, we define a new region of interest for this values as the interval $[4, 6]$. Here, the new region of interest reads as follows:

$$S0 \in [2, 3], A \in [1.05, 2], G \in [20, 80].$$

Latin hypercube sampling with 100 points and three repeats each is used for a graphical exploration of the ROI. Figure 14.8 displays a fit of the response surface which is based on the complete data set (320 runs of the target algorithm). A local regression model based on R's `loess()` function is fitted to the data.

Summarizing Results from the RSM

Results from this case study support results presented by Beyer (2001). The factor a should be chosen in the range from 1.1 to 1.5. Further experiments show that this

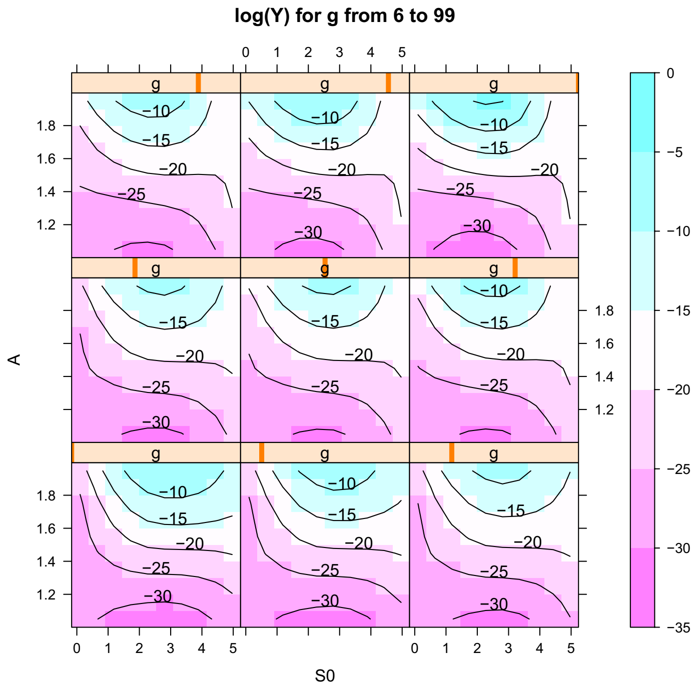

Fig. 14.8: Contour plot based on the complete data set (ES-Sphere with 320 function evaluations). Smaller values are better. Better configurations are placed in the lower-left corner of the panels. A is plotted versus $S0$, while values of the factor with the last but one effect, namely G, are varied with the slider on top of each panel

result is independent of the starting point and problem dimension. The corresponding contour plots indicate that runs of the (1+1)-ES with values for a chosen in the recommended interval result in good function values. The structure of the contour plots does not change if starting point or problem dimension are modified.

14.7 Additional Model Considerations

The analysis of optimization algorithms requires the investigation of categorical variables; e.g., in evolution strategies, several variants of the recombination operator can be used (Beyer and Schwefel 2002). SPOT allows the coding of nonnumerical variables as factors. *Dummy regressors* or *contrasts* can be used to represent levels of a factor. SPOT can handle the following variables:

1. real values,
2. integers, and

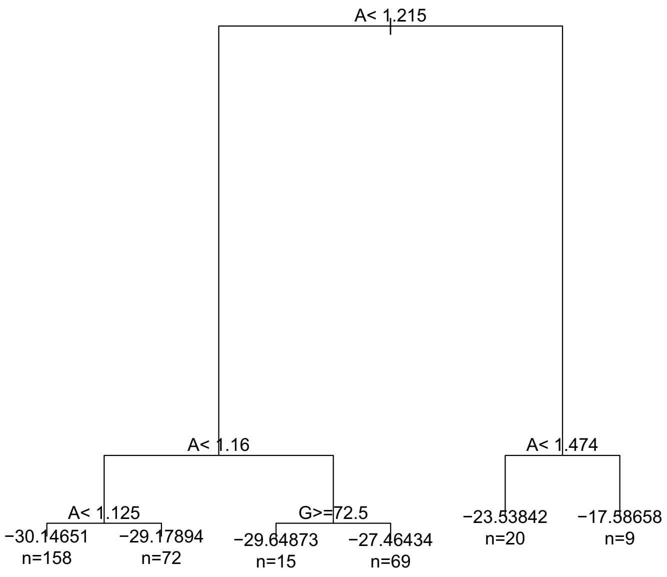

Fig. 14.9: Tree-based regression. The same data as in Fig. 14.8 were used. The factor A has the largest effect. $\log(y)$ values were used to grow the tree

3. categorical variables.

SPOT's ROI file is used to specify this typing. Maindonald and Braun (2003) illustrate the treatment of dummy variables for classical regression in R.

Classical regression is only one technique that can be used in the SPOT framework to predict interesting design points. Alternatively, *tree-based models* can be used to cope with categorical variables; see also Fig. 14.9. Tree-based methods can be used for regression as well as for classification (Breiman et al. 1984). Maindonald and Braun (2003) recommend to use tree-based methods in tandem with parametric approaches, because tree-based regression may suggest interaction terms that ought to appear in a parametric model. However, tree-based methods require more data than classical regression techniques. That is, if only a few runs of the algorithms are possible, it may be necessary to use parametric models. On the other hand, tree-based methods may be helpful to explore new data sets very quickly: the experimenter gets on overview of which variables have the greatest effects on the algorithm's performance.

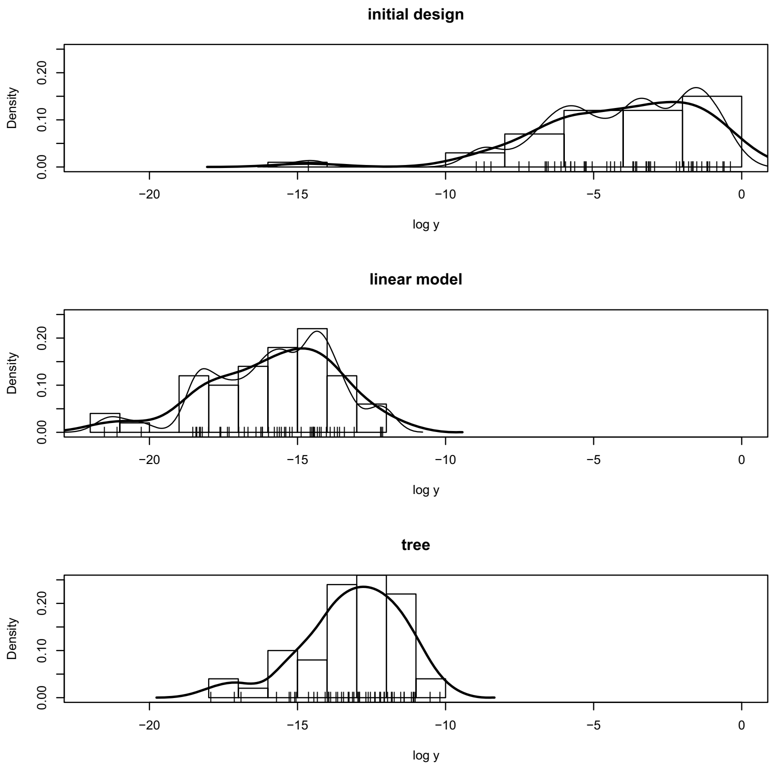

Fig. 14.10: Comparison of two SPOT prediction models with randomly generated configurations (from *top* to *bottom*). First, the results from randomly generated algorithm designs are shown. We have chosen the initial design (LHD), which is generated randomly. Next, results from 50 repeats of the best algorithm design determined with the linear and the tree-based regression model are displayed

We have mentioned only two prediction models: parametrized regression and regression trees. Further models, e.g., local regression, Kriging or mixed models as introduced in Sect. 10.1, are also available.

Example 14.2. To analyze the choice of the prediction model on the prediction quality and SPOT's ability to improve optimization algorithms (tuning), we performed the following study: Results from two different SPOT runs are compared. The first run uses a tree-based regression model, whereas in the second run the dummy-variable regression model was used. Both models used exactly the same setting, i.e., five SPOT iterations, initial design size of 50 points, and two repeats. An ES, which optimizes the ten-dimensional sphere function with 1,000 function evaluations was analyzed. Distributions of the results from these two SPOT runs are compared with the distribution of the randomly generated initial design in Fig. 14.10. Both models were able to find better results than the randomized design. These results are in line with observations from other studies: Tree-based models can be applied very easily to unknown explanatory variables. They can cope with categorical and numerical data. Parametrized models perform better than tree-based models; however, the costs for modeling are higher. □

14.8 The Automated Mode

In the case study in Sect. 14.6 we discussed the initial setup for the SPOT loop (initial design) and the analysis of one step. SPOT can proceed as follows: Based on the prediction model, e.g., linear regression or tree-based regression, interesting algorithm design points are generated. These design points are evaluated, i.e., the algorithm is run with the corresponding parameters. Then, an analysis as described in Sect. 14.6.2 can be performed. This analysis provides an improved predictor, which can be used to propose new design points, and so forth. As depicted in Fig. 14.2, this procedure can be performed in an automated manner. The result from the automated approach reads

$$s_0 = 4.99, a = 1.10, g = 71.$$

Obviously, the result from the automated approach supports the findings from the manual approach, i.e., similar values for the parametrization of the (1+1)-ES are determined.

14.9 Summary

In this chapter, basic elements of the SPOT framework were introduced and discussed. SPOT is an open-source implementation of the sequential parameter optimization presented in Chap.2. It requires the specification of the region of interest, the algorithm design, and SPOT-related configuration parameters.

SPOT provides tools to perform the following four elementary tasks: (i) generate an initial design, (ii) run the algorithm, (iii) generate a new design based on results from previous runs, and (iv) generate a report. Additionally, these tasks can be run in an automated mode.

We demonstrated how a simple analysis of the (1+1)-ES can be performed. This analysis requires the specification of the following elements:

1. Region of interest, i.e., the range of parameter settings of the algorithm.
2. Algorithm- and problem-related parameters.
3. Metaparameters used by SPOT.

A simple regression model, well known in DOE and RSM (Montgomery 2001), was used to analyze the influence (importance) of three parameters of the (1+1)-ES, namely, initial stepsize, multiplier for the stepsize, and the length of the memory vector. The experimental results support a result which was derived theoretically by Beyer (2001).

SPOT is being developed, applied, and improved at several research institutes around the world. By providing an open-source implementation and a graphical user interface, we hope that SPOT can be a useful tool for the research community.

Acknowledgements This work has been supported by the Bundesministerium für Forschung und Bildung (BMBF) under the grant FIWA (AIF FKZ 17N2309, "Ingenieurnachwuchs") and by the Cologne University of Applied Sciences under the grant COSA.

References

Armitage P (1975) Sequential medical trials, 2nd edn. Blackwell, Oxford, U.K.

Bartz-Beielstein T (2010a) Sequential parameter optimization. Tech. Rep. 04/2010, Institute of Computer Science, Faculty of Computer Science and Engineering Science, Cologne University of Applied Sciences, Germany, URL `http://www.gm.fh-koeln.de/imperia/md/content/personen/lehrende/bartz_beielstein_thomas/spotannotatedbib.pdf`

Bartz-Beielstein T (2010b) SPOT: An R package for automatic and interactive tuning of optimization algorithms by sequential parameter optimization. Tech. Rep. CIOP TR 05-10, Cologne University of Applied Sciences, URL `http://arxiv.org/abs/1006.4645`, related software can be downloaded from http://cran.r-project.org/web/packages/SPOT/index.html

Bartz-Beielstein T, Lasarczyk C, Preuß M (2005) Sequential parameter optimization. In: McKay B, et al. (eds) Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland, IEEE Press, Piscataway NJ, vol 1, pp 773–780

Beyer HG (2001) The Theory of Evolution Strategies. Springer

Beyer HG, Schwefel HP (2002) Evolution strategies—A comprehensive introduction. Natural Computing 1:3–52

Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and Regression Trees. Wadsworth, Monterey CA

Chen J, Chen C, Kelton D (2003) Optimal computing budget allocation of indifference-zone-selection procedures, working paper, taken from `http://www.cba.uc.edu/faculty/keltonwd`. Cited 6 January 2005

Draper NR, Smith H (1998) Applied Regression Analysis, 3rd edn. Wiley, New York NY

Fober T, Mernberger M, Klebe G, Hüllermeier E (2009) Evolutionary construction of multiple graph alignments for the structural analysis of biomolecules. Bioinformatics 25(16):2110–2117

Fox J (2002) An R and S-Plus Companion to Applied Regression. Sage

Ihaka R, Gentleman R (1996) R: A language for data analysis and graphics. Journal of Computational and Graphical Statistics 5(3):299–314

Kleijnen JPC (1987) Statistical Tools for Simulation Practitioners. Marcel Dekker, New York NY

Konen W, Zimmer T, Bartz-Beielstein T (2009) Optimierte Modellierung von Füllständen in Regenüberlaufbecken mittels CI-basierter Parameterselektion Optimized Modelling of Fill Levels in Stormwater Tanks Using CI-based Parameter Selection Schemes. at-Automatisierungstechnik 57(3):155–166

Lasarczyk CWG (2007) Genetische Programmierung einer algorithmischen Chemie. PhD thesis, Technische Universität Dortmund

Lophaven S, Nielsen H, Søndergaard J (2002) DACE—A Matlab Kriging Toolbox. Tech. Rep. IMM-REP-2002-12, Informatics and Mathematical Modelling, Technical University of Denmark, Copenhagen, Denmark

Maindonald J, Braun J (2003) Data Analysis and Graphics using R—an Example-based Approach. Cambridge University Press, Cambridge UK

Mehnen J, Michelitsch T, Bartz-Beielstein T, Henkenjohann N (2004) Systematic analyses of multi-objective evolutionary algorithms applied to real-world problems using statistical design of experiments. In: Teti R (ed) Proceedings Fourth International Seminar Intelligent Computation in Manufacturing Engineering, CIRP ICME'04, Naples, Italy, vol 4, pp 171–178

Mehnen J, Michelitsch T, Lasarczyk C, Bartz-Beielstein T (2007) Multi-objective evolutionary design of mold temperature control using DACE for parameter optimization. International Journal of Applied Electromagnetics and Mechanics 25(1–4):661–667

Montgomery DC (2001) Design and Analysis of Experiments, 5th edn. Wiley, New York NY

Pukelsheim F (1993) Optimal Design of Experiments. Wiley, New York NY

Rudolph G, Preuss M, Quadflieg J (2009) Two-layered surrogate modeling for tuning optimization metaheuristics. Algorithm Engineering Report TR09-2-005, Faculty of Computer Science, Algorithm Engineering (Ls11), Technische Universität Dortmund, Germany

Santner TJ, Williams BJ, Notz WI (2003) The Design and Analysis of Computer Experiments. Springer

Tukey J (1991) The philosophy of multiple comparisons. Statistical Science 6:100–116

Volkert L (2006) Investigating ea based training of hmm using a sequential parameter optimization approach. In: Yen GG, Lucas SM, Fogel G, Kendall G, Salomon R, Zhang BT, Coello CAC, Runarsson TP (eds) Proceedings of the 2006 IEEE Congress on Evolutionary Computation, IEEE Press, Vancouver, BC, Canada, pp 2742–2749, URL `http://ieeexplore.ieee.org/servlet/opac?punumber=11108`

Yi Y (2008) Fuzzy operator trees for modeling utility functions. PhD thesis, Philipps-Universität Marburg

Ziegenhirt J, Bartz-Beielstein T, Flasch O, Konen W, Zaefferer M (2010) Optimization of biogas production with computational intelligence—a comparative study. Tech. Rep. 03/2010, Institute of Computer Science, Faculty of Computer Science and Engineering Science, Cologne University of Applied Sciences, Germany