An abstract interface for surrogate optimization in the PLATON framework

Abul K.M Fahimuddin*, Markus Krosche, Hermann G. Matthies

Institute of Scientific Computing, Technical University of Braunschweig, Hans-Sommer-Strasse 65, Braunschweig, 38106, Germany

Abstract

We describe an abstract interface for surrogate-model based optimization in the PLATON framework. The abstract interface encapsulates data structure and ensures easy integration of various third party implementation of surrogate methods. Kriging, a statistical method originating from the field of geostatistics, is used here to replace the expensive simulation code. The paper concentrates on the software development issues related to the integration of surrogate modeling in PLATON. In particular, PLATON's flexible structure for easy incorporation of third-party implementations is portrayed. At the end, a numerical experiment is provided for justifying the realization of the PLATON-Surrogate framework.

Keywords: Component-based software engineering; Numerical optimization; Abstract interface; PLATON; Surrogate modeling; Kriging

1. Introduction

Computer-based simulation and analysis are used extensively in engineering design and multidisciplinary design optimization (MDO). There exists a distinct discrepancy between the computational cost of complex high-fidelity engineering simulations and the growth of computing power and speed. One way to address this challenge is to use the approximation methods. The basic idea is to construct a simplified mathematical approximation of the computationally expensive simulation and analysis code, which is then used in place of the original code. Since the approximation model acts as a surrogate for the original code, it is often referred to as a surrogate model or a meta model (i.e. a model of a model).

In this paper, we explore one of the approximation methods called kriging. The use of an abstract interface in the PLATON [1] framework helps to build robust and easily extensible codes. Also PLATON ensures computational steering of the optimization processes. The central focus of this paper lies on the development of an appropriate interface for surrogate optimization in PLATON. The paper is organized as follows: the next section describes the PLATON system. Section 3 elaborates the surrogate modeling and kriging. In section 4, we describe the abstract interface for kriging in PLATON. Finally, we finish with a numerical example and conclusions.

* Corresponding author. Tel.: +49 (531) 391 3000; Fax: +49 (531) 391 3003; E-mail: a.fahimuddin@tu-bs.de

2. The PLATON framework

In component-based software engineering (CBSE), units of software are encapsulated as 'components', which interact with other components only through well-defined interfaces, see Larson et al. [2]. This approach allows the internal implementation of the component to remain opaque to the rest of the world, and presumably hides much of the complexity of the software. The PLATON framework is inspired by the CBSE principle. PLATON is developed for distributed numerical optimization. Figure 1 shows a schematic view of the PLATON framework. Here the optimizers and simulators are coupled through the communication middleware CTL (Communication Template Library). PLATON has an abstract view of the function that is to be minimized by determining the appropriate parameters. For further details on PLATON, we refer to Krosche et al. [1].

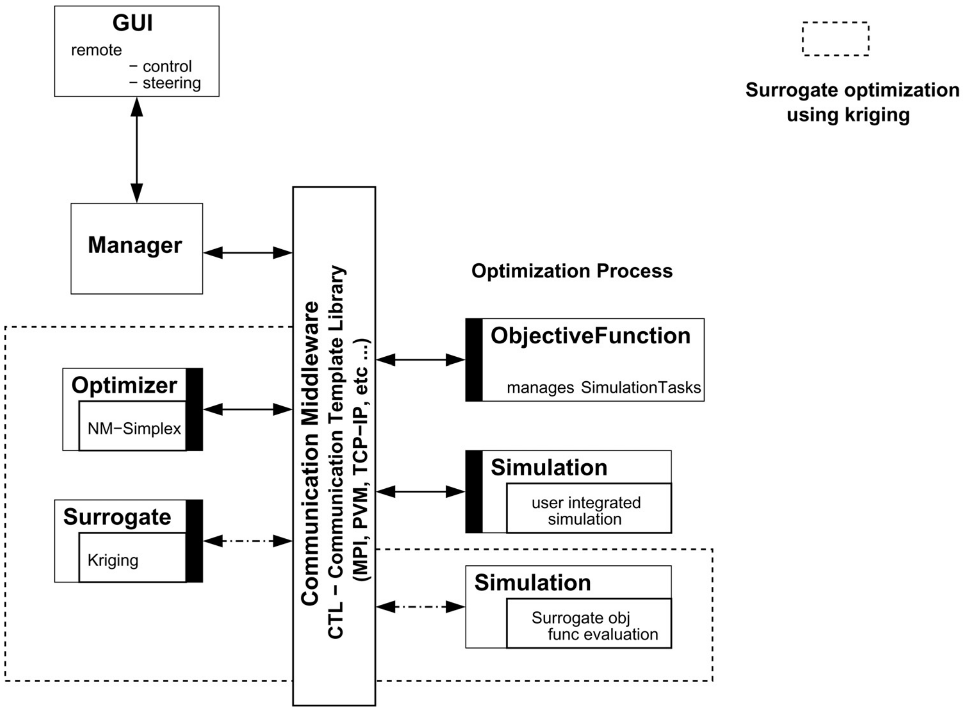

Fig. 1. PLATON software framework [3].

3. Numerical optimization using surrogate models

The basic approach to construct a simplified mathematical approximation of the computationally expensive simulation and analysis code is to approximate the original code with a reliable surrogate model. A variety of approximation methods exist (e.g. polynomial response surfaces, kriging models, radial basis functions, neural networks, multivariate adaptive regression spline, etc.).

Mathematically, the general optimization problem can be stated as the following:

$$\textbf{minimize} \quad f(x)$$
$$\text{subject to} \quad x \in \mathcal{B} \subseteq \mathcal{R}^d$$

where $f : \mathcal{R}^d \to \mathcal{R} \cup \{\infty\}$. In the present case, it is assumed that the evaluation of the objective function $f(x)$ requires running expensive analysis code(s). And hence, the inexpensive surrogate objective functions $\hat{f}(x)$ shall be used to accelerate for a solution without sacrificing theoretical guarantees of asymptotic convergence, see Booker et al. [4]. Two of the most important surrogate models are response surface modeling (RSM) and kriging. The availability of a third-party implementation has prompted us to use kriging in this work.

3.1. Construction of surrogates using kriging

Kriging [5] is a weighted average of known function values, where weights are chosen based on the locations of known function values, in order to minimize the error of prediction. The mathematical model of kriging is based on a stationary random function. Let us consider the case of n sampled points $x^1, x^2, \ldots, x^n$ in a d-dimensional design space at which the system behavior $y(x)$ has been evaluated. The corresponding responses at the sample points are denoted by $\boldsymbol{Y} = (y_1, \ldots, y_n)^T$. Following the approach of Schonlau [6], the kriging approximation expresses the response as a function of the independent design variables. The kriging approximation is expressed as the linear combination of a number of selected known functions (i.e. regression models) and a stochastic process:

$$\boldsymbol{Y}(x) = \sum_{j=1}^{k} \beta_j \boldsymbol{f}_j\,(x) + \boldsymbol{Z}(x) \tag{1}$$

where $\boldsymbol{f}_j\,(x)$ are k known regression models, β_j are the corresponding parameters, and $\boldsymbol{Z}\,(x)$ is a stochastic process with mean zero and variance σ^2. Here $\boldsymbol{Z}(x)$ creates a localized deviation so that the kriging model interpolates the n sampled data points. In many kriging applications, including this work, a simpler model is used,

$$\boldsymbol{Y}(x) = \beta + \boldsymbol{Z}\,(x) \tag{2}$$

The covariance between the $\boldsymbol{Z}$'s at two design points x^1 and x^2 is given by:

$$\text{Cov}\,[\boldsymbol{Z}\,(x^i), \boldsymbol{Z}\,(x^j)] = \sigma^2 \boldsymbol{R}([\mathcal{R}(x^i, x^j)]) \tag{3}$$

In the above formulation $\boldsymbol{R}$ is the correlation matrix and $\mathcal{R}(x^i, x^j)$ is the correlation function between any two of the n sampled data points x^i and x^j.

A variety of correlation functions exist and here we use the Gauss correlation function:

$$\mathcal{R}(x^i, x^j) = \exp\left[-\sum_{k=1}^{d} \theta_k |x_k{}^i - x_k{}^j|^{p_k}\right] \tag{4}$$

where θ_k and p_k are the unknown correlation parameters used to fit the model, $x_k{}^i$ and $x_k{}^j$ are the k th components of sample points. Here $\theta_k \geq 0$ and $0 < p_k \leq 2$. The parameter p_k can be interpreted as an indicator of increasing the smoothness of the response surface, while larger θ_k indicates greater nonlinearity.

The kriging method solves an optimization problem through the application of nonlinear mathematical programming to an estimated solution space, see Sakata et al. [7]. A flowchart of this design process is shown in Fig. 2. For details on the kriging method, especially the computation of the parameters p_k and θ_k which is known as the maximum likelihood estimation, we refer to Fahimuddin [3].

4. Concepts of abstract interfaces

Component-based software engineering (CBSE) is concerned with the assembly of pre-existing software components into larger pieces of software. Underlying this process is the notion that software components are

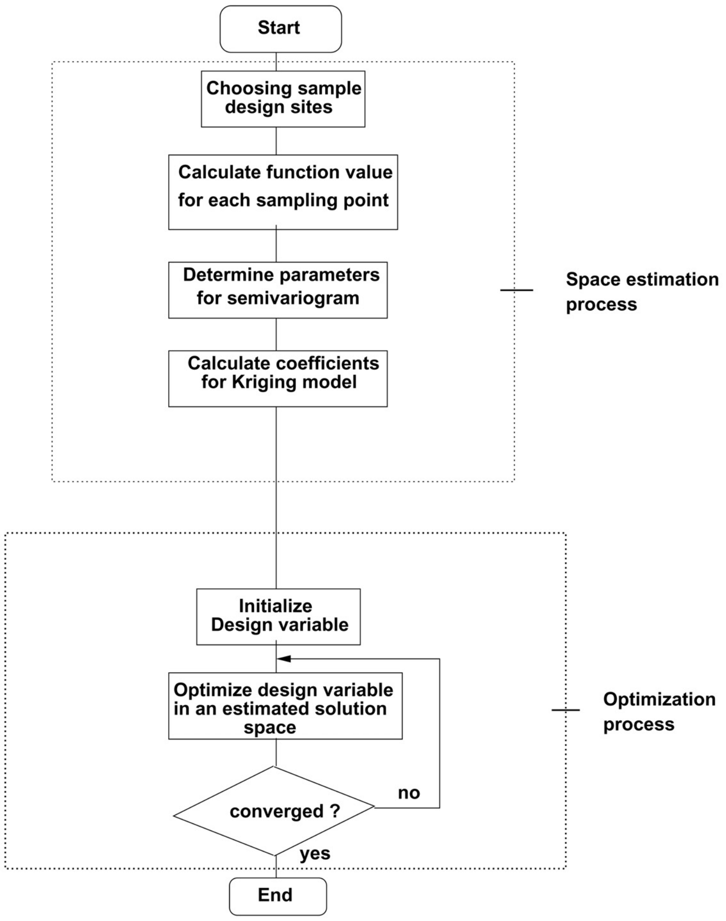
Start
Choosing sample design sites
Calculate function value for each sampling point
Determine parameters for semivariogram
Calculate coefficients for Kriging model
Space estimation process
Initialize Design variable
Optimize design variable in an estimated solution space
converged ?
no
yes
Optimization process
End

Fig. 2. Optimization using kriging solution space, see Sakata et al. [7].

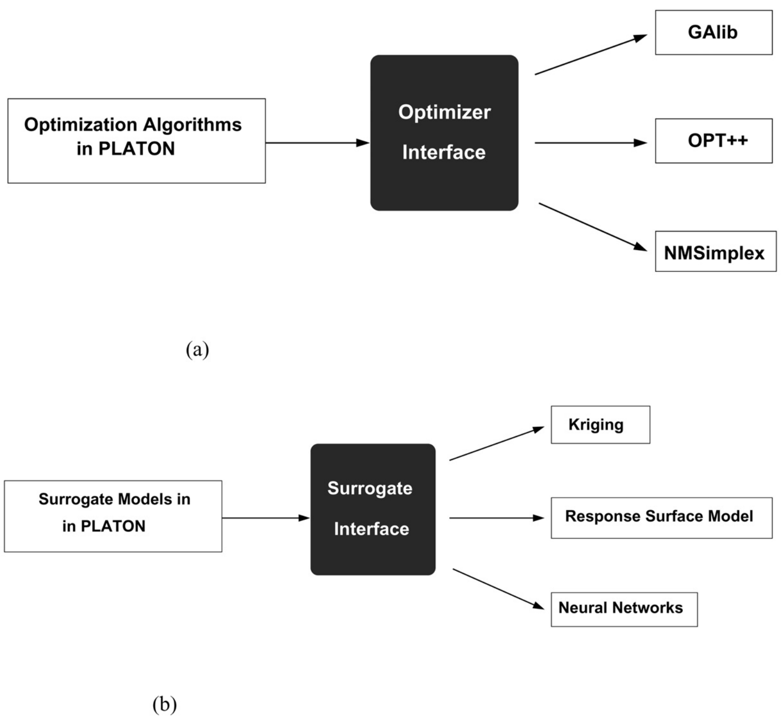

Fig. 3. Data structure encapsulation with abstract interfaces in PLATON. (a) Abstract interface for optimizers in PLATON. (b) Abstract interface for surrogate models in PLATON.

written in such a way that they provide functions common to many different systems. Ideally, a component is a black box; its services are only accessible through a well-defined interface. Figure 3(a) shows how we can use various optimizers in PLATON through interfaces. Because of the *Optimizer* interface, the user can choose among the available optimizers, e.g. GAlib, OPT++, NMSimplex, etc. The application scientists can use the desired optimizer with minimum programming effort. Because of its abstract nature, it is very easy to modify or add new optimizers by simply manipulating the interface.

4.1. PLATON–Surrogate abstract interface

PLATON provides a flexible architecture to insert abstract interfaces for optimization and simulation and hence it is chosen as the platform to implement the surrogate interface. Figure 3(b) shows the proposed *Surrogate* interface for kriging and response surface modeling. Once the abstract interface is declared, we are in a position to choose among the available meta models. In the present case, we use kriging as the desired surrogate method. In order to be consistent with the philosophy of third party software usage in PLATON, we have decided to use the kriging subroutines developed by Padula et al. [8]. The object orientation features, i.e. polymorphism, inheritance, etc., of the original code enable us to easily integrate it in PLATON. This kriging routine is accessed through the surrogate interface in PLATON, see Fig. 1.

4.2. Implementation of the PLATON–Surrogate interface

In designing the abstract interface for surrogate models, we must follow the coding styles and appropriate function calling methods of PLATON. In the following listing, a glimpse of the interface for the kriging model is given. The important feature of this code segment is its abstract nature, i.e. it is independent of the type of surrogate method used. The methods used in this interface are the minimum for constructing any meta model. The *init*() method initializes the whole optimization process in PLATON, while the *setEvaluator*() is the simulation component inside PLATON. The core method *buildModel*() calls the third party implementation of kriging. The *evaluate*() function evaluates the objective function on the estimated kriging surface; *setProperty*() implements the specific properties of the meta model. The specification of model parameters are performed through XML files. As a result of this

Listing 1: *Abstract interface for surrogate modeling in PLATON*

```
#ifndef __SURROGATE_CI
#define __SURROGATE_CI
#include <platon.ci>
#include <typemap.hpp>
#define CTL_Library modeling
#include CTL_LibBegin
 #define CTL_Class Surrogate
 #include CTL_ClassBegin
  #define CTL_Method1 void,init,(const utl::int4, const utl::int4,const de::tubs::wire::TypeMapT),3
  #define CTL_Method2 void,setEvaluator,(const platon::Evaluator), 1
  #define CTL_Method3 void, buildModel,(), 0
  #define CTL_Method4 void, improveModel,(), 0
  #define CTL_Method5 utl::any, evaluate,(const utl::vector<utl::real8 >),1
  #define CTL_Method6 void,setProperty,(const std::string,const utl::any),2
 #include CTL_ClassEnd
#include CTL_LibEnd
#endif // __SURROGATE_CI
```

abstract nature of integration, this interface can easily be extended for other meta models, e.g. response surface modeling. The dotted rectangle in Fig. 1 portrays the whole idea of this surrogate modeling.

5. Numerical example and performance analysis

In order to clarify the concept of the previously described surrogate based optimization in the PLATON framework, we have used Rosenbrock's function as a benchmark problem. It is a function of two variables, $g(x, y) = (1 - x)^2 + 105(y - x^2)^2$ which has multiple minima and a global minimum of 0 at the point (1,1).

We used the proposed PLATON-Surrogate framework to implement a kriging approximation of Rosenbrock's function which is used as the objective function. Figure 4 portrays the consecutive kriging approximations of the original Rosenbrock function. We compute the global minimum of this approximated surface. In this case, according to Padula et al. [8], a simple search has been employed to find this minimum. After a few steps, we obtain the minimum at the point (1,1). At this stage, we have identified the design parameters of the minimum, but the model does not reproduce the correct function value (it predicts -3.43382 instead of 0).

The accuracy of the meta models like kriging depends on the validation tests, e.g. cross validation. In this example, no additional efforts have been made for determining the accuracy of the kriging model as this is not the goal of this work. Also the Gaussian isotropic correlation function is used here; we could use some other functions, e.g. a cubic correlation function on the unit cube or the Matérn correlation function. For maximum likelihood estimation, the Direct Search algorithm has been used; we could replace this with other optimization methods. The later version of the PLATON-Surrogate framework will incorporate the above mentioned features for fine tuning of the kriging model parameters.

6. Conclusion

In this paper, we have demonstrated the use of the concepts of an abstract interface for surrogate modeling in the PLATON framework. The main focus has been put on the integration of a third party kriging implementation into PLATON. The numerical results are promising, but a further investigation is necessary for the kriging modeling itself to improve the overall optimization process.

References

[1] Krosche M, Niekamp R, Matthies HG. PLATON: A problem solving environment for computational steering of evolutionary optimisation on the grid. In: Proc. of the 5th Conference on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems, Barcelona, Spain, G Bugeda, JA Desideri, J Periaux, M Schoenauer, G Winter EUROGEN 2003.

[2] Larson JW, Norris B, Ong ET et al. Components, the common component architecture, and the climate/weather/ocean community. In: Proc of the 20th International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography and Hydrology Seattle, DC 2004.

[3] Fahimuddin AKM. Analysis and implementation of concepts for the coupling of global optimization methods and DACE in PLATON framework. Master's thesis, TU Braunschweig, Germany, 2004.

[4] Booker AJ, Dennis JE, Frank PD, Serafini DB, Torczon V, Trosset MW. A rigorous framework for optimization of expensive functions by surrogates. Struct Optimization 1999;17:1–13.

[5] Simpson TW, Mauery TM, Korte JJ, Mistree F. Kriging metamodels for global approximation in simulation-based

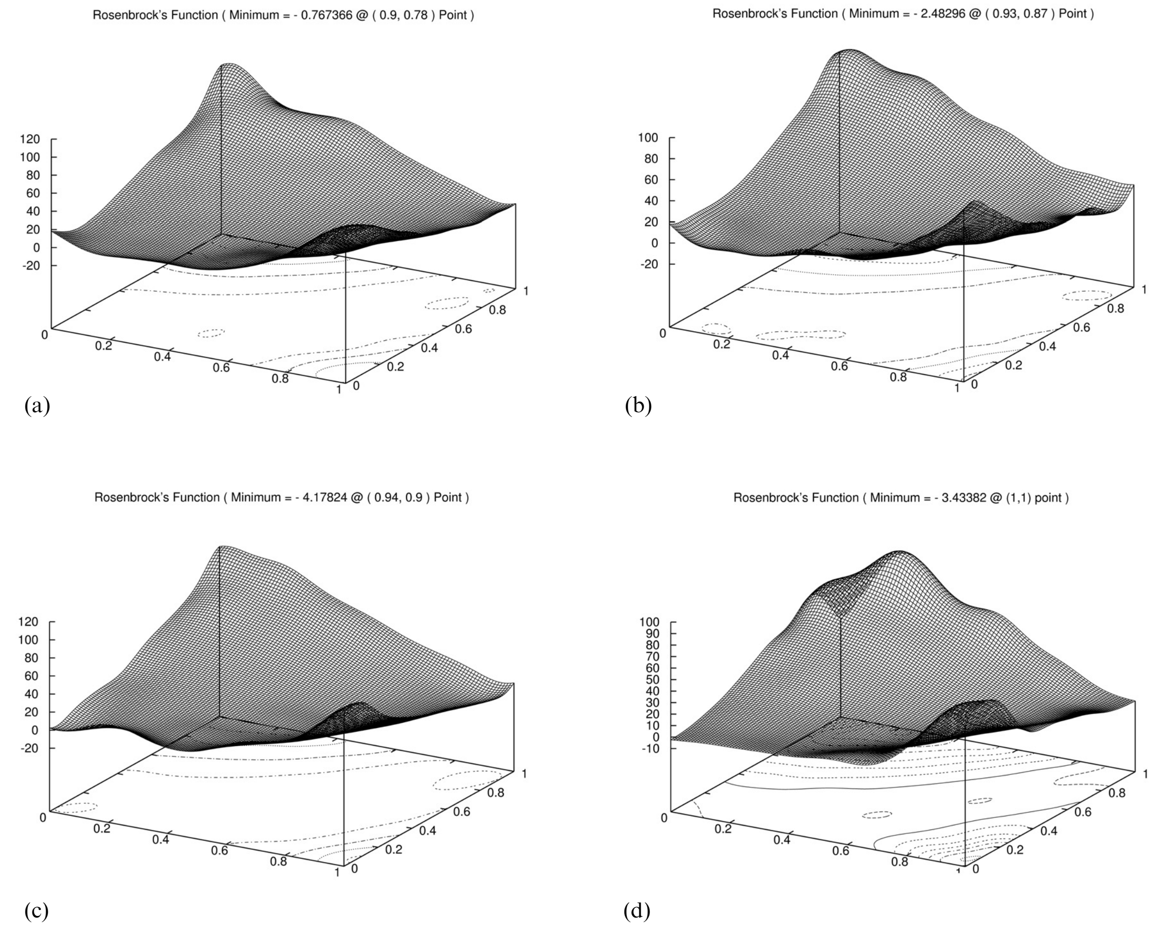

Fig. 4. Approximated solution space by kriging in the PLATON–Surrogate framework. (a) Estimated surface: phase 1; (b) estimated surface: phase 2; (c) estimated surface: phase 3; (d) estimated surface: phase 4.

multidisciplinary design optimization. AIAA J 2001;39:2233–2241.

[6] Schonlau M. Computer experiments and global optimization. Ph.D. dissertation, University of Waterloo, 1997.

[7] Sakata S, Ashida F, Zako M. Structural optimization using kriging approximation. Comput Methods Appl. Mech. Eng 2003; 192(7–8):923–939.

[8] Padula A, Trosset MW, Torczon V. Krigifier and Kriging Approximation Classes. http://www.cs.wm.eud/∼va/software/krigifier, 2003.