Engineering Design via Surrogate Modelling

A Practical Guide

Alexander I. J. Forrester, András Sóbester and Andy J. Keane

University of Southampton, UK

WILEY

A John Wiley and Sons, Ltd., Publication

Engineering Design via Surrogate Modelling

Engineering Design via Surrogate Modelling

A Practical Guide

Alexander I. J. Forrester, András Sóbester and Andy J. Keane

University of Southampton, UK

WILEY

A John Wiley and Sons, Ltd., Publication

This edition first published 2008

Registered office
John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom

For details of our global editorial offices, for customer services and for information about how to apply for permission to reuse the copyright material in this book please see our website at www.wiley.com.

Library of Congress Cataloging in Publication Data

Forrester, Alexander I. J.
Engineering design via surrogate modelling : a practical guide / Alexander I.J. Forrester, András Sóbester, and Andy J. Keane.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-470-06068-1 (cloth : alk. paper) 1. Engineering design—Mathematical models. 2. Engineering design—Statistical methods.
I. Sóbester, András. II. Keane, A. J. III. Title.
TA174.F675 2008
620′.0042015118—dc22
2008017093

A catalogue record for this book is available from the British Library

ISBN 978-0-470-06068-1

Set in 10/12pt Times by Integra Software Services Pvt. Ltd. Pondicherry, India
Printed and bound in Great Britain by Antony Rowe Ltd, Chippenham, Wiltshire

Contents

Preface

Think of a well-known public personality whom you could easily identify from a photograph. Consider now whether you would still recognize them if most of the photograph was obscured, except for the corner of an eye, a small part of their chin and, perhaps, a half of their mouth. This is a game often played on television quiz shows and some contestants (and viewers at home) often display an uncanny ability to come up with the correct name after only a few small sections of the picture are revealed.

This is a demonstration of the brain's astounding ability to fill in blanks by subconsciously constructing a *surrogate model* of the full photograph, based on a few samples of it. The key to such apparently impressive feats is that we actually know a great deal about the obscured parts. We know that the photograph represents a human face, that is the image is likely to be roughly symmetrical, and we know that somewhere in the middle there must be a pattern we usually refer to as a 'nose', etc. Moreover, we know that it is a famous face. The 'search space' thus reduced, the task seems a lot easier.

The surrogate models that form the subject of this book are educated guesses as to what an engineering function might look like, based on a few points in space where we can afford to measure the function values. While these glimpses alone would not tell us much, they become very useful if we build a number of assumptions into the surrogate based on our experience of what such functions tend to look like. For example, they tend to be continuous. We may also assume that their derivatives are continuous too. With such assumptions built into the learner, the surrogate model becomes a very effective low cost replacement of the original function for a wide variety of purposes.

Surrogate modelling has had a great impact on the way the authors think about design and, after many years of combined experience in the subject, it has become a fundamental element of our engineering thought processes. We wrote this book as a means of sharing some of this experience on a practical level. While a lot has been written about the deeper theoretical aspects of surrogate modelling (indeed, references are included throughout this text to the landmarks of this literature that have informed our own thinking), what we strove to offer here is a manual for the practitioner wishing to get started quickly on solving their own engineering problems. Of course, like any sharp tool, surrogate modelling can only be used in a scientifically rigorous way if the user is constantly aware of its dangers, pitfalls, potential false promises and limitations – the present text goes to great lengths to point these out at the appropriate times.

To emphasize the practical dimension of this guide, we accompany it with our own *MATLAB*® implementation of the techniques described therein. Snippets of this code are included in the text wherever we felt that, through the 'maths-like' and compact nature of *MATLAB*, they contribute to the explanations. These, as well as all the rest of the code, can be found on the book website at `www.wiley.com/go/forrester`. Template scripts are also provided, ready for the user to replace our objective function modules with his or her own. It is worth noting here that our own example functions, while mostly representing 'real life' engineering problems, were designed for easy experimentation; that is they take only fractions of a second to run. We expect, however, that most of the applications the codes will be used for 'in anger' will be several orders of magnitude more time-consuming.

This is a self-contained text, though we assumed a basic familiarity with calculus, linear algebra and probability. Additional 'mathematical notes' are included wherever we had to refer to more advanced topics within these subjects. We therefore hope that this book will be useful to graduate students, researchers and professional engineers alike.

While numerous colleagues have assisted us in the writing of this volume, a few names stand out in particular. We would like to thank Prasanth Nair and David Toal of the University of Southampton, Max Morris of Iowa State University, Donald Jones of the General Motors Co., Natalia Alexandrov of NASA, Tom Etheridge of Astrium, Lucian Tudose of the Technical University of Cluj Napoca, Danie G. Krige and Stephen J. Leary of BAE Systems for their suggestions and for reading various versions of this manuscript.

Finally, a disclaimer. Surrogate modelling is a vast subject and this text does not claim nor, indeed, can hope to cover it all. The selection of techniques we have chosen to include reflect, to some extent, our personal biases. In other words, this is the combination of tools that works for us and we earnestly hope that it will for the reader too.

Alexander Forrester, András Sóbester and Andy Keane
Southampton, UK

Disclaimer

About the Authors

Dr Alexander I. J. Forrester is Lecturer in Engineering Design at the University of Southampton. His main area of research focuses on improving the efficiency with which expensive analysis (particularly computational fluid dynamics) is used in design. His techniques have been applied to wing aerodynamics, satellite structures, sports equipment design and Formula One.

Dr András Sóbester is a Lecturer and EPSRC/Royal Academy of Engineering Research Fellow in the School of Engineering Sciences at the University of Southampton. His research interests include aircraft design, aerodynamic shape parameterization and optimization, as well as engineering design technology in general.

Professor Andy J. Keane currently holds the Chair of Computational Engineering at the University of Southampton. He leads the University's Computational Engineering and Design Research Group and directs the Rolls-Royce University Technology Centre for Computational Engineering. His interests lie primarily in the aerospace sciences, with a focus on the design of aerospace systems using computational methods. He has published over two hundred papers and three books in this area, many of which deal with surrogate modelling concepts.

Foreword

Over the last two decades, there has been an explosion in the ability of engineers to build finite-element models to simulate how a complex product will perform. In the automotive industry, for example, we can now simulate the injury level of passengers in a crash, the vibration and noise experienced when driving on different road surfaces, and the vehicle's life when subjected to repeated stressful conditions such as pot holes. Moreover, our ability to quickly modify these simulation models to reflect design changes has greatly increased. The net result is that the potential for using optimization to improve an engineering design is now higher than ever before.

One of the major obstacles to the use of optimization, however, is the long running time of the simulations (often overnight) and the lack of gradient information in some of the most complicated simulations (especially crashworthiness). Due to the long running times and the lack of analytic gradients, almost any optimization algorithm applied directly to the simulation will be slow.

Despite this slowness, one could still bite the bullet and invest one's computational budget in applying an optimization algorithm directly to the simulations. But this is unlikely to be satisfying, because rarely does a single optimization result settle any design issue. For example, if the result is *not* satisfactory, one may want to gain insight into what is going on by performing parameter sweeps and plotting input-output relationships. Or one might want to repeat the optimization with a modified formulation (different starting point, different constraints). All this, of course, requires doing more simulations. On the other hand, if the result *is* satisfactory, one still want might to do further investigations to see if a better tradeoff can be struck between competing objectives. Again, this requires more simulations. Clearly, if one uses up all the available resources solving the first optimization problem, all these follow-up studies would not be possible, or at least lead to missed deadlines.

The basic idea in the 'surrogate model' approach is to avoid the temptation to invest one's computational budget in answering the question at hand and, instead, invest in developing fast mathematical approximations to the long running computer codes. Given these approximations, many questions can be posed and answered, many graphs can be made, many tradeoffs explored, and many insights gained. One can then return to the long running computer code to test the ideas so generated and, if necessary, update the approximations and iterate.