



DSDApp: An Open-Access Tool for Definitive Screening Design

SOFTWARE
METAPAPER

RYOICHIRO HAYASAKA

JENS HÄNISCH

PABLO CAYADO

*Author affiliations can be found in the back matter of this article

ubiquity press

ABSTRACT

Definitive Screening Design (DSD) is a powerful design of experiment technique employed to find significant factors (parameters) and optimize the factor levels in many experiments of different fields. However, making its design and analyzing the obtained results usually requires expensive software or advance programming knowledge. Therefore, the authors of this work developed a web application, called “DSDApp”, with which users can make designs and analyze the results of DSD without statistical and programming efforts. DSDApp can generate DSD tables, make second-order models, and optimize factor levels to obtain max/min/target output values.

CORRESPONDING AUTHORS:

Ryoichiro Hayasaka

Karlsruhe Institute of Technology (KIT), Institute for Technical Physics (ITEP), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
long.hayasaka.6832@gmail.com

Pablo Cayado

University of Geneva,
Department of Quantum Matter Physics (DQMP), Quai Ernest-Ansermet 24, 1211 Geneva, Switzerland
pablo.cayado@unige.ch

KEYWORDS:

Design of Experiment (DOE); Definitive Screening Design (DSD); Multi-objective Optimization

TO CITE THIS ARTICLE:

Hayasaka R, Hänisch J, Cayado P 2024 DSDApp: An Open-Access Tool for Definitive Screening Design. *Journal of Open Research Software*, 12: 2. DOI: <https://doi.org/10.5334/jors.462>

(1) OVERVIEW

INTRODUCTION

Design of experiments (DOE) includes various statistical approaches for planning, executing and analysing experiments in a systematic and efficient way. Definitive Screening Design (DSD) [1] is a particularly useful DOE technique in fields such as material engineering [2, 3, 4], chemistry [5, 6] and biology [7, 8], where numerous factors are involved in the experiments. The DSD has significant advantages over conventional DOEs. Most conventional DOE techniques are for experiments with two levels (e.g. $-1/+1$) for all factors. If there is a second-order effect of factor A (i.e. AA) that deals with two levels, a conventional DOE cannot find an optimal condition that may exist between the two levels. Even though some conventional DOEs can deal with three levels (e.g., $-1/0/+1$), such as L18 orthogonal array, the second-order effects often involve correlations with other factor effects. This implies that ambiguity may remain in the process of building models. In contrast, DSD deals with three levels (low, middle, and high values) of the factors (parameters) and the correlations between the factors are minimized. These features are useful for building second-order models and finding the optimal levels of factors in the investigated parameter windows of the experiments.

However, designing and analyzing DSD experiments typically requires commercial software (e.g., JMP, Design-Expert and Minitab) or programming knowledge (e.g., R or Python), which is, in many cases, a barrier for researchers to employ DSD in their experiments. Considering this situation, the authors of this work developed DSDApp, a free web application that provides an effortless way to use DSD for researchers with a lack of access to commercial software packages or statistical programming knowledge.

The user-friendly interface of DSDApp allows users to design DSD experiments, to create second-order models (including main effects, two-factor interactions and quadratic effects) and to perform parameter optimization

to obtain desirable objective values. The app can be used in a very simple way only by button-clicking, which makes DSDApp a more attractive and easier option to employ DSD than existing R packages (e.g., “daewr”) or Python libraries (e.g., “definitive-screening-design”) that require statistical programming experience.

DSDApp is particularly profitable for researchers in small research or academic organizations or companies without access to commercial software. Also, those without any previous experience with DOE techniques or with limited experience in the use of DSD may benefit from the app because of its simpler and more intuitive operation compared to commercial software that offer complicated functionalities for more specialized needs. Overall, DSDApp is expected to extend the use of DSD and broaden experiments in various research and development fields.

IMPLEMENTATION AND ARCHITECTURE

As shown in Figure 1, DSDApp offers three main functionalities: planning definitive screening designs, making regression models, and optimizing parameters. The detailed usage of DSDApp is explained in the following sections.

PLANNING DSD

In the “Plan” tab, one can create a DSD table with 4 to 12 factors. The process should start by changing the levels (low: -1 , middle: 0 , high: 1) of the factors. These levels are corresponding to the minimum, middle, and maximum values of the factors. Figure 2 shows an example of a six-factor (A, B, C, D, E, F) DSD table generated by DSDApp. Each row is an experimental run that users should execute with the factor levels given, and collect result Y. According to Ref. 9, two or more “fake factors” should be considered when constructing a DSD table for better effect detection performance. Fake factors do not correspond to the actual variables (therefore no need to consider when doing experiments), but only act when building a model as described in Appendix.

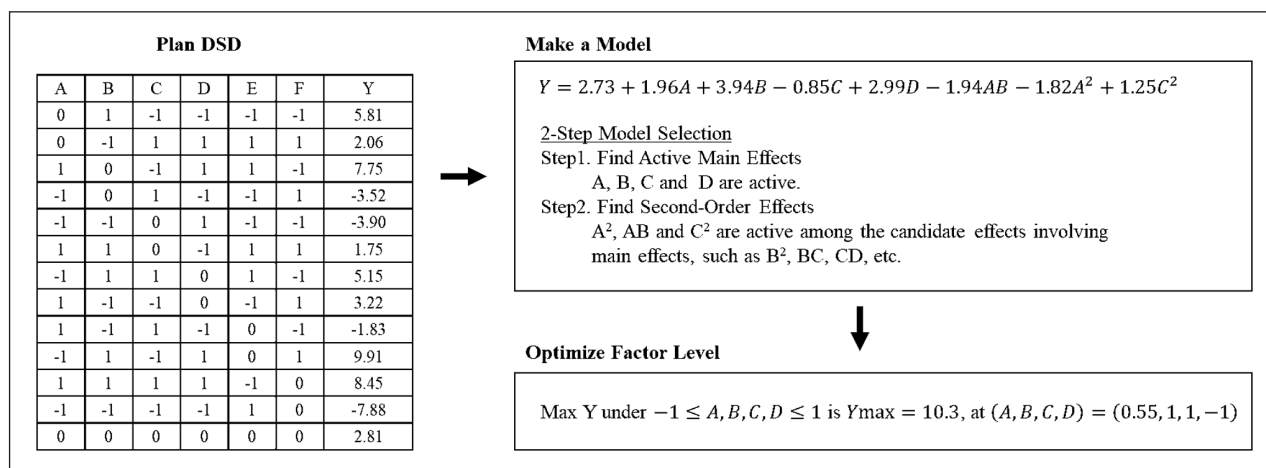


Figure 1 Overview of DSDApp.

The table generated can be downloaded as a csv file by clicking on “Download.” In the experiment, the actual data should be recorded and added in a new column in the downloaded csv file.

In Figure 2, instead of actual data, the checkbox “Generate the sample data” was ticked to generate sample data given by:

$$y = 3 + 2A + 4B - C + 3D - 2AA - 2AB + CC + \varepsilon, \varepsilon \sim N(0, \sigma = 0.3) \quad (1)$$

This model will be used for testing the app in the section “Quality control”.

MAKING MODELS

Before making the models, it is necessary to upload the file with the DSD table and experimental result(s). Multiple objective variables can be included in the added columns. By clicking on “Browse” (Figure 3), it is possible to upload a csv/txt file in the DSDApp from your computer.

The models can be built by following the steps below.

1. Set the result column as an objective variable “Y”, input variables (A, B, C, D) as “X”, and fake factors (E, F) as “Fake Factor.”
2. Click on “Find active terms” to obtain active main factors and second-order factors (Step 1). The detection of main and second-order factors is described in the tab “Step 1” as shown in Figure 4.
3. After pressing “Find active terms,” the possible first- and second-order terms appear in “X1” and “X2” tab automatically, following the model building strategy detailed in the Appendix. The user can consider which terms should be included in the model and manually add/remove factors of interest (Step 2).

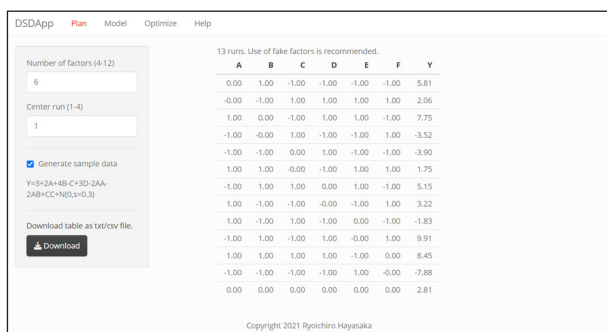


Figure 2 Planning of DSD. By clicking “Download” in the left bottom corner, users can download the table shown in the right.

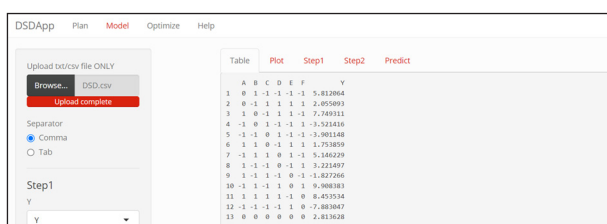


Figure 3 Uploading of DSD and experiment result.

4. (Optional) If the user manually altered terms in “X1”, press “Regenerate X2” to regenerate terms in “X2” by combining the first-order terms selected in “X1.” This procedure is optional if the user is satisfied with the terms that automatically appeared in “X1” and “X2” after pressing “Find active terms”.
5. Click on “Build model” to make a model with the terms “X1” and “X2.” The summary of the model is described in the tab “Step2” as shown in Figure 5.

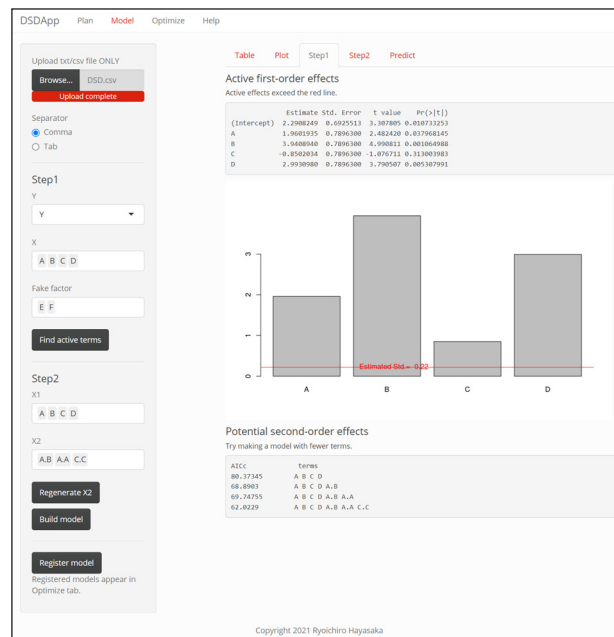


Figure 4 Step 1: Detecting main factors and second-order terms.

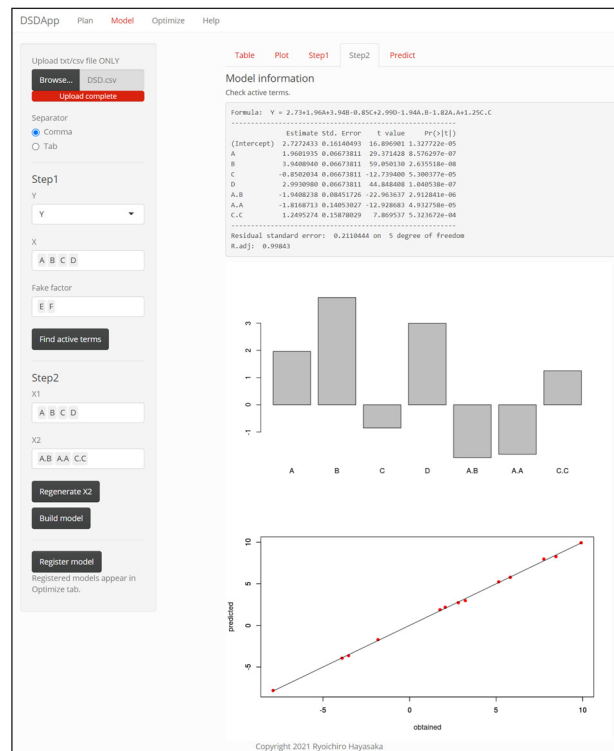


Figure 5 Step 2: Model making based on terms X1 and X2. The model information and the bar graph represent the coefficients of the terms. The red scatter plot displays how well the model can explain the obtained values.

DSDApp employs a model-selection strategy tailored for DSD, where active main (first-order) factors are detected first and then second-order terms related to main factors are included [9]. For example, when main factors A and B are active (or set as X1), possible second-order terms (X2) are AA, AB and BB. To avoid overfitting, the second-order terms are included to minimize Akaike Information Criteria with finite correction (AICc). See the Appendix for a more detailed explanation.

After creating the model, the prediction of the output value is possible in the tab “Predict”. The input vector \mathbf{x} (or the factor levels) can be set to specified values, as it can be seen in Figure 6. The prediction value \mathbf{y}_{x_0} at $\mathbf{x}_0 = [1, A, B, C, D, \dots]$ and its prediction interval is calculated as:

$$y_{x_0} \pm t_{\alpha/2, n-p} \sqrt{\sigma^2 (1 + \mathbf{x}_0 (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{x}_0)} \quad (2)$$

where \mathbf{X} is the design matrix of DSD, α is the significance level (0.05), n is the number of runs, p is the number of terms in the model (including the intercept term) and $t_{\alpha/2, n-p}$ is t -value with two-sided confidential level α and the degree of freedom $n-p$.

OPTIMIZING FACTOR LEVEL

DSDApp offers the possibility to optimize the parameters, i.e., to find the optimal set of parameters for the main factors to obtain the desired objective variable(s). DSDApp transforms the objective value y into a “desirability function” that evaluates how satisfactory y is on a scale from 0 “not satisfactory” to 1 “completely satisfactory.” Figure 7 illustrates the desirability functions for three different cases: minimization, maximization and tuning y at a specific value. For minimizing and maximizing, the individual desirability function D_1 is expressed as:

$$D_1 = \left[1 + 99 \exp \left\{ \left(y - \frac{y_{\text{allowable}} + y_{\text{target}}}{2} \right) \left(\frac{2p}{y_{\text{allowable}} - y_{\text{target}}} \right) \right\} \right]^{-1}, \quad (3)$$

$$p = \begin{cases} -1 & (\text{for minimizing}) \\ 1 & (\text{for maximizing}) \end{cases}$$

where $y_{\text{allowable}}$ refers to the minimum required value that the user wishes to secure, and y_{target} refers to the ideal value. In the case of minimization, for example, if $y > y_{\text{allowable}}$, D_1 gets smaller than 0.01 as shown in Figure 7(a). This value of D_1 is small enough to consider y as unsatisfactory when $y > y_{\text{allowable}}$ and, therefore, y tends to be smaller than $y_{\text{allowable}}$. In the case of maximization, the value of D_1 gets smaller when $y < y_{\text{allowable}}$ as shown in Figure 7(b), and thus y tends to be larger than $y_{\text{allowable}}$. Note that y cannot get much smaller (or larger) than y_{target} because D_1 remains stable when $y < y_{\text{target}}$ for minimization, and $y > y_{\text{target}}$ for maximization.

Figure 7(c) shows the individual function D_2 for tuning at a certain level y_{target} . The function D_2 was expressed as:

$$D_2 = \begin{cases} \exp \left\{ -\frac{(y - y_{\text{target}})^2}{2} \left(\frac{y_{\text{target}} - y_{\text{lower}}}{3} \right)^{-2} \right\}, & (y \leq y_{\text{target}}) \\ \exp \left\{ -\frac{(y - y_{\text{target}})^2}{2} \left(\frac{y_{\text{upper}} - y_{\text{target}}}{3} \right)^{-2} \right\}, & (y > y_{\text{target}}) \end{cases} \quad (4)$$

The desirability functions are calculated for various sets of the factor levels \mathbf{x} (e.g., $\mathbf{x} = [-0.5, +0.2, 0, \dots, 1]$). The set \mathbf{x} that maximizes the desirability is the optimal parameter set. For the optimization calculation, limited-memory quasi-Newton code for bound-constrained optimization (L-BFGS-B) is employed (the function “optim” in R language can do the optimization).

In the case of having multiple objective variables, multi-objective optimization is also possible. Instead of maximizing individual desirability, the total desirability

$$D_{\text{total}} = \left(\prod_{i=1}^n D_i \right)^{1/n} \quad (5)$$

is maximized; where i is the iterator corresponding to n objective variables. If no parameter set can meet all the limitations, i.e., if some of the desirability functions are zero, then D_{total} is evaluated as 0. In such situation, broadening a bit the limitations ($y_{\text{allowable}}$) of some objective variables can be helpful to obtain an optimization result.

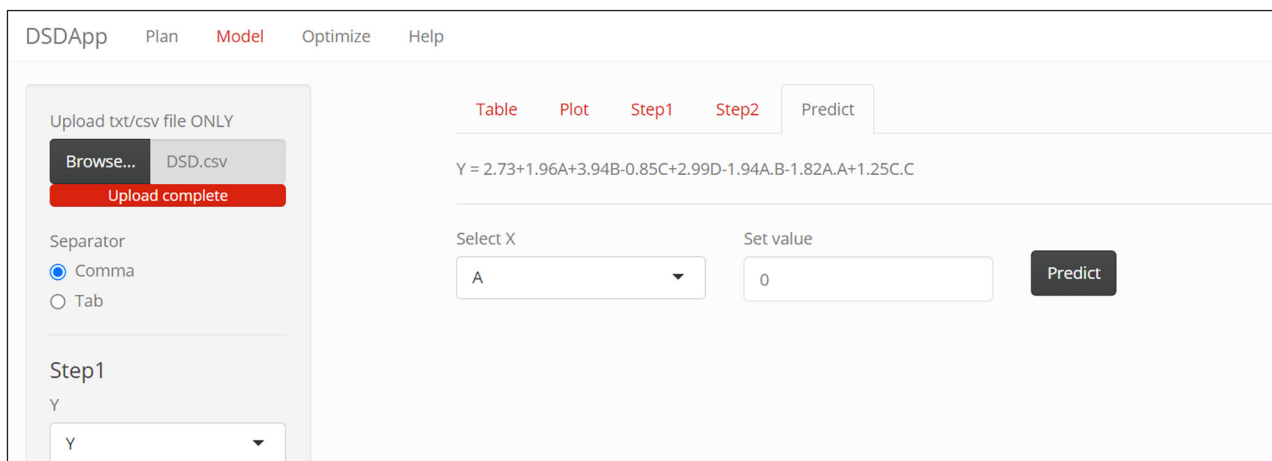


Figure 6 Prediction of output variable at specified input variables.

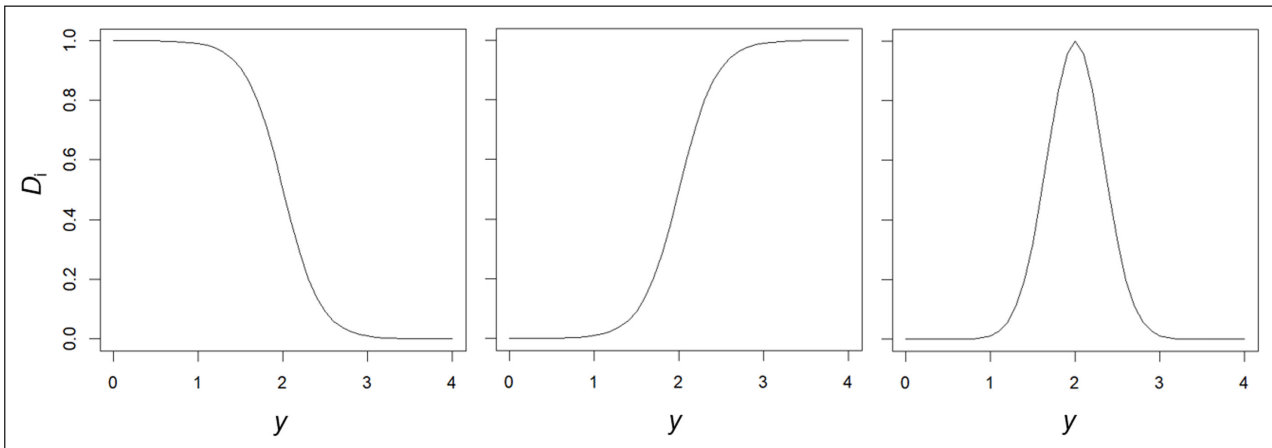


Figure 7 Desirability functions; (a) minimization, (b) maximization, and (c) tuning (at $y_{target} = 2$) with $y_{lower} = 1$ and $y_{upper} = 3$.

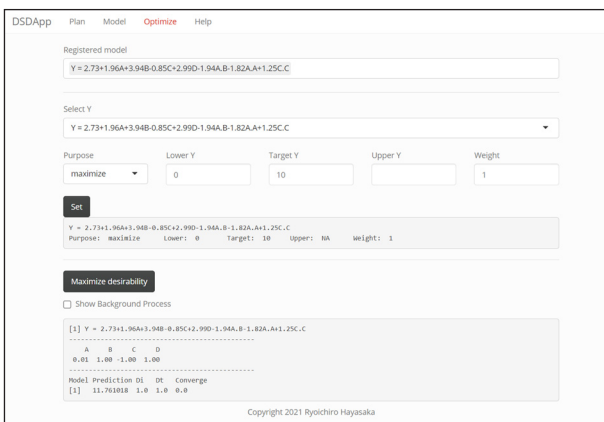


Figure 8 Optimization.

The optimization procedure is as follows.

1. Click on “Register model” as shown in Figure 5. The registered model shows up in the selector in Figure 8.
2. Click on “Set” button to define the purpose (minimize/maximize/target) and lower or raise the limits.
3. Click on “Maximize desirability” to optimize total desirability D_{total} . “Maximize desirability” needs to be clicked several times because optimization starts from different initial points and can lead to different optimal conditions.
4. For multiple output values, it is necessary to register all the models and set their purposes of optimization individually.

QUALITY CONTROL

DSDApp has been validated by using a six-factor DSD table and simulated observations. The DSD table includes columns A-D as real factors and E and F as fake factors as shown in Table 1 (same as the table in Figure 5). Then, we generated simulated data y using the predefined models (same as the equation in Figure 2).

$$y = 3 + 2A + 4B - C + 3D - 2AA - 2AB + CC + \varepsilon, \varepsilon \sim N(0, \sigma = 0.3) \quad (1).$$

NO.	A	B	C	D	E	F	Y
1	0	1	-1	-1	-1	-1	5.812
2	0	-1	1	1	1	1	2.055
3	1	0	-1	1	1	-1	7.749
4	-1	0	1	-1	-1	1	-3.521
5	-1	-1	0	1	-1	-1	-3.901
6	1	1	0	-1	1	1	1.754
7	-1	1	1	0	1	-1	5.146
8	1	-1	-1	0	-1	1	3.221
9	1	-1	1	-1	0	-1	-1.827
10	-1	1	-1	1	0	1	9.908
11	1	1	1	1	-1	0	8.454
12	-1	-1	-1	-1	1	0	-7.883
13	0	0	0	0	0	0	2.814

Table 1 Data and simulated result.

The users can verify the app functionality by checking whether or not the model built based on observation y is similar to the predefined model (1).

Afterwards, the model below was built based on the generated data using DSDApp as shown in “Model Information” in Figure 5.

$$y = 2.886 + 2.018A + 4.014B - 0.917C + 2.904D - 1.968AB - 1.9355AA + 1.106CC \quad (6)$$

As shown in Figure 9, the coefficients of the built model (6) and the original model (1) are similar, but with slight differences. To confirm that these differences are negligible compared to the variation ε , another set of the input points was introduced and the prediction validity of the model was checked.

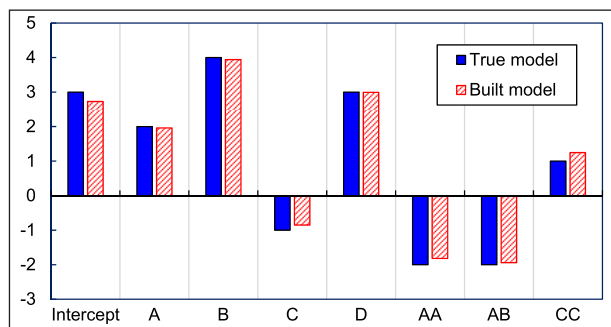


Figure 9 Comparison between the true model (1) in Table 2 and the built model.

A	B	C	D	Y	\hat{Y}
1	-1	-1	-1	0	-0.02245
-1	1	-1	-1	4	3.938109
1	1	-1	-1	4	3.977689
-1	-1	1	-1	-10	-9.52573
-1	1	1	-1	2	2.237702
1	1	1	-1	2	2.277282
-1	-1	-1	1	-2	-1.83913
1	-1	-1	1	6	5.963744
1	1	-1	1	10	9.963885

Table 2 Confirmation points (A, B, C, D). y is generated by the original function (see Test 1 in Table 3), and \hat{y} is generated by the model (9).

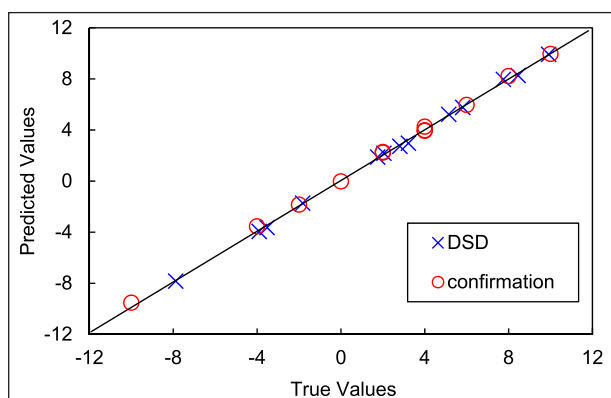


Figure 10 Predicted and true values for DSD points (blue) and confirmation points (red) in Table 5. The fact that all the points align on the black line shows that the prediction agrees well with the original values.

Table 2 shows the confirmation points at the edge of the experimental space, where the maximum prediction error is expected to be observed. Figure 10 shows the correlation between predicted values \hat{y} based on the model (6) (vertical axis) and the true values y generated by the original function (horizontal axis) for the same input parameters as in Table 2. The blue dots correspond to y and \hat{y} at the DSD points,

and the red dots to the confirmation points. By using the values of y and \hat{y} , the residual sum of squares for the confirmation points is calculated as 0.156, which is smaller than the original standard error $\sigma = 0.30$. Thus, the constructed model (6) fits well with the true model (1). Nevertheless, one should be cautious when several quadratic effects and two-factor interactions are active because these effects in DSD sometimes have relatively strong correlations [1]. In such a case, several possible models should be evaluated in later experiments to see which model is the most useful and reliable.

Similar tests in this section can be done by users' defined equations. For example, "test.R" generates CSV files (DSD8-with-Y1.csv, DSD8-with-Y2.csv) with a DSD table and sample data by a hand-made equation. Users can upload the generated CSV files in DSDApp, and then check if the built models are similar to user's predefined models.

(2) AVAILABILITY

OPERATING SYSTEM

Windows (tested on Windows 10)

PROGRAMMING LANGUAGE

R 3.6.3 (R.4.0)

ADDITIONAL SYSTEM REQUIREMENTS

Users can access to DSDApp at https://my-first-dsd.shinyapps.io/DSDApp_ver2/ on a web browser app.

For local usage, Rstudio is needed. Open server.R or ui.R in Rstudio, then click on "Run App."

DEPENDENCIES

R packages daewr, shiny, and shinythemes.

LIST OF CONTRIBUTORS

P.C. conceptualized the idea and, together with J.H., provided supervision. R.H. developed the app and carried out tests to check its correct operation. All authors were involved in the writing of the original and the revised manuscript.

SOFTWARE LOCATION

Code repository

Name: GitHub

Identifier: <https://github.com/long-rh/DSDApp>

Licence: MIT License

Date published: 03/12/2022

LANGUAGE

English

LITERATURE	MODELS OR ACTIVE TERMS IN LITERATURE	MODELS OR ACTIVE TERM IN DSDAPP	REMARKS
[5]	$y = 56.1 + 8.9B - 9.3C + 21.0G + 16.8H - 9.0$	$y = 56.1 + 8.9B - 9.3C + 21.0G + 16.8H - 9.1$	F and I were set as fake factors in DSDApp.
[7]	$y = 1170 + 10A + 216E^* - 316G^* - 181H^* - 83I - 169AE^* - 313HH^* + 81AH + 17D + 85DD$	$y = 1000 + 10.2A + 216E^* - 316G^* - 180H^* - 83I - 153AxE^* - 214HH^* + 97AH - 98EG$	B and C were set as fake factors, and A was manually included as a main factor in DSDApp. “*” was marked as significant in the literature.
[1]	A, B, C, AB, AA	A, B, C, AB, AA	No fake factors

Table 3 The validation of DSDApp based on literature data.

(3) REUSE POTENTIAL

DSDApp, accessible at https://my-first-dsd.shinyapps.io/DSDApp_ver2/, is designed for researchers and experimenters working in various fields who wish to include DSD in the experimental routine. Its intuitive interface allows users to navigate the application easily through button clicks.

The app is suitable for numerous applications where efficient and systematic experimentation is crucial. For example, in material engineering [3, 4], factors such as solution concentration, heating temperature, and humidity can be correlated with each other and need to be optimized for superior material properties.

Although specific examples of DSDApp usage have not been provided in literature, its potential applications are evident in the wide range of fields that rely on DSD as part of the experimental process. In Table 3, three examples are provided to show the potential of the use of the DSDApp on data already available in literature. For each reference, the DSD condition table and results were uploaded in DSDApp, the model generated in the app and compared to the one in the literature. The first example [5] shows that the models in literature and the app are almost the same. The models in the second example [7] were slightly different, but the terms with * (marked as significant in the literature) were successfully identified as active in DSDApp, too. In the final example [1], although the coefficients of the model were not provided in the ref. [1] the same active terms were identified in DSDApp and in the publication. Note that the models based on experimental results can be built by different model-selection strategies [10], which causes differences in the resulting models. Also, the terms that should be included in the model depend on the knowledge and experience of the researcher in the field. Therefore, the models in literature and DSDApp are not always the same. The users have to keep in mind that DSD employs the specific model-selection strategy explained in detail in the Appendix.

Future development of DSDApp will include the ability to perform mixed-level DSD with two- and three-factors, allowing experimenters to incorporate blocking factors

or categorical factors. This enhancement will further expand the app’s applicability to researchers who require more complex experimental designs.

For questions about using the app or inquiries regarding its application to specific research areas, R.H. and P.C. can be contacted via email. They can provide guidance and support to any user that wants to use the DSDApp for their experiments.

ADDITIONAL FILE

The additional file for this article can be found as follows:

- **Appendix.** Model-selection strategy in DSDApp. DOI: <https://doi.org/10.5334/jors.462.s1>


ACKNOWLEDGEMENTS


The initial experimental work for developing DSDApp has been conducted at Karlsruhe Institute of Technology (Germany) during the first author’s research visit within the program COLABS offered by Tohoku University (Japan). The author thanks the research members at The Institute for Technical Physics and Tohoku University for giving insight into this work.


COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR AFFILIATIONS

Ryoichiro Hayasaka  orcid.org/0000-0003-1160-3846
 Karlsruhe Institute of Technology (KIT), Institute for Technical Physics (ITEP), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany

Jens Hänisch  orcid.org/0000-0003-2757-236X
 Karlsruhe Institute of Technology (KIT), Institute for Technical Physics (ITEP), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany

Pablo Cayado  orcid.org/0000-0003-3703-6122
University of Geneva, Department of Quantum Matter Physics
(DQMP), Quai Ernest-Ansermet 24, 1211 Geneva, Switzerland

REFERENCES

1. **Jones B, Nachtsheim CJ.** A class of three-level designs for definitive screening in the presence of second-order effects. *Journal of Quality Technology*. 2011; 43(1): 1–15. DOI: <https://doi.org/10.1080/00224065.2011.11917841>
2. **Libbrecht W, Deruyck F, Poelman H, Verberckmoes A, Thybaut J, De Clercq J, Van Der Voort P.** Optimization of soft templated mesoporous carbon synthesis using Definitive Screening Design. *Chemical Engineering Journal*. 2015; 259: 126–134. DOI: <https://doi.org/10.1016/j.cej.2014.07.113>
3. **Rijckaert H, Cayado P, Hänisch J, Billet J, Erbe M, Holzapfel B, Van Driessche I.** Unravelling the Crystallization Process in Solution-Derived YBa₂Cu₃O_{7-δ} Nanocomposite Films with Preformed ZrO₂ Nanocrystals via Definitive Screening Design. *Journal of Physical Chemistry Letters*. 2021; 12(8): 2118–2125. DOI: <https://doi.org/10.1021/acs.jpcllett.1c00135>
4. **Hayasaka R, Cayado P, Erbe M, Freitag W, Hänisch J, Holzapfel B, Ito S, Hashizume H.** Investigation of the crystallization process of CSD-ErBCO on IBAD-substrate via DSD approach. *Scientific Reports*. 2020; 10(1). DOI: <https://doi.org/10.1038/s41598-020-76848-y>
5. **Fidaleo M, Lavecchia R, Petrucci E, Zuorro A.** Application of a novel definitive screening design to decolorization of an azo dye on boron-doped diamond electrodes. *International Journal of Environmental Science and Technology*. 2016; 13(3): 835–842. DOI: <https://doi.org/10.1007/s13762-016-0933-3>
6. **Van Heugten AJP, Braal CL, Versluijs-helder M, Vromans H.** European Journal of Pharmaceutical Sciences. *The influence of cetomacrogol ointment processing on structure: A definitive screening design*. 2017; 99: 279–284. DOI: <https://doi.org/10.1016/j.ejps.2016.12.029>
7. **Tai M, Ly A, Leung I, Nayar G.** Efficient high-throughput biological process characterization: Definitive screening design with the Ambr250 bioreactor system. *Biotechnology Progress*. 2015; 31(5): 1388–1395. DOI: <https://doi.org/10.1002/btpr.2142>
8. **Beyecha K, Abdissa D.** Heliyon Optimization of biodiesel production parameters from Prosopis julifera seed using definitive screening design. *Heliyon*. 2022; 8(November 2021): e08965. DOI: <https://doi.org/10.1016/j.heliyon.2022.e08965>
9. **Jones B, Nachtsheim CJ.** Effective Design-Based Model Selection for Definitive Screening Designs. *Technometrics*. 2017; 59(3): 319–329. DOI: <https://doi.org/10.1080/00401706.2016.1234979>
10. **Burnham KP, Anderson DR.** Multimodel Inference. *Sociological Methods & Research*. 2004; 33(2): 261–304. DOI: <https://doi.org/10.1177/0049124104268644>

TO CITE THIS ARTICLE:

Hayasaka R, Hänisch J, Cayado P 2024 DSDApp: An Open-Access Tool for Definitive Screening Design. *Journal of Open Research Software*, 12: 2. DOI: <https://doi.org/10.5334/jors.462>

Submitted: 17 March 2023 **Accepted:** 06 February 2024 **Published:** 21 February 2024

COPYRIGHT:

© 2024 The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC-BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. See <http://creativecommons.org/licenses/by/4.0/>.

Journal of Open Research Software is a peer-reviewed open access journal published by Ubiquity Press.

Appendix: Model-selection strategy in DSDApp

In the framework of DSD, possible second-order models are created after experimenting with the chosen design. One of the common ways of building models is the use of the Akaike information criterion with finite correction (AICc). AICc (or generally AIC) is an estimator to select a “good” model that can explain the given data and avoid overfitting by including as few terms as possible in the model. Assuming that the errors of all factors follow independent and identical normal distributions $N(0, \sigma^2)$ with variance σ^2 , AICc is expressed for the least square estimation[10] as:

$$\text{AICc} = n \ln(\hat{\sigma}^2) + 2k + \frac{2k(k+1)}{n-k-1} \quad (1) ,$$

where n is the number of runs and k is the number of the factors in the model. Here, the estimated variance $\hat{\sigma}^2$ is calculated by:

$$\hat{\sigma}^2 = \frac{\sum (y_i - \hat{y}_i)^2}{n} \quad (2) ,$$

where y_i and \hat{y}_i are observed and predicted values, respectively, of the i^{th} factor. As can be seen from equation (1), a “good” model has a smaller value of AICc because $\hat{\sigma}^2$ should be small, and k should be also small to avoid overfitting. In DSDApp, the two-step approach similar to the effective model selection in reference[9] is employed. Algorithm 1 describes how the model is selected in DSDApp.

First of all, the error s_e is estimated based on either or both of the N_c center runs and N_f fake factors. The error based on the center runs is calculated by:

$$s_c = \sqrt{\frac{1}{N_c - 1} \sum_{i=1}^{N_c} (y_i - \bar{y}_c)^2} \quad (3) ,$$

and the error based on the fake factors:

$$s_f = \sqrt{\frac{Y^t P_F Y}{N_F}}, \quad P_F = X_F (X_F^t X_F)^{-1} X_F^t \quad (4) ,$$

where X_F is the matrix of the fake factors. In the case of Table 1, the rows and columns of X_F consist of those of fake1 and fake2. Considering both s_c and s_f , the overall error s_e is estimated by:

$$s_e = \sqrt{\frac{(N_c - 1)s_c^2 + s_f^2}{N_c + N_f - 1}} \quad (5) .$$

Note that the further procedure is not performed if there are no further center runs or no fake factors.

Based on observation Y (the vector of obtained data $[y_1, \dots, y_n]^t$), the first-order regression model \hat{Y}_{1st} is built. If the coefficient of a factor in the model \hat{Y}_{1st} exceeds the error s_e multiplied by the t -value for an 80% confidence interval, the factor is included as one of the main factors. Then, the

first-order model $\widehat{Y}_{\text{main}}$ is made by using these main factors. Afterwards, the active quadratic terms are selected. Here, the quadratic terms are comprised of the main factors. This is to say that if factors A and B are included in the main factors, AA, BB, and AB are the candidate terms. Figure 1 shows the model selection process in the case of the main factors A and B. To select the effective quadratic terms, the candidate term is evaluated by forward regression with the initial model $\widehat{Y}_{\text{main}}$ adding the term whose p-value is minimum among the candidate. The same procedure is continued for the model with added quadratic terms until no more effective quadratic terms (with p-value <0.2) are found. Each time a quadratic term is added, the aforementioned AICc is calculated for the model, and the model with minimum AICc is selected as the final model. A more detailed explanation of this algorithm is given by reference[9].

Algorithm 1

- 1) Estimate error s_e (see equations (9) and (10)).
- 2) Calculate b_i ($i=1, \dots, k$), the coefficients of the first-order regression model $\widehat{Y}_{1\text{st}} = \mathbf{bX}$.
- 3) If $b_i / t(N_c + N_f - 1, 0.8) > s_e$, add i^{th} factor into main factors.
- 4) Make a first-order model $\widehat{Y}_{\text{main}}$ using the main factors.
- 5) Make a candidate group of second-order terms comprised of main factors.
- 6) Select effective quadratic terms among the candidate group.
 - I. Add quadratic terms by forward regression with the initial model $\widehat{Y}_{\text{main}}$.
 - II. Calculate AICc for the added model.
 - III. Stop adding quadratic terms if the p-value of the added term exceeds 0.2.
 - IV. Select the final model with minimum AICc
- 7) Make a multi-regression model by using main factors and effective quadratic terms.

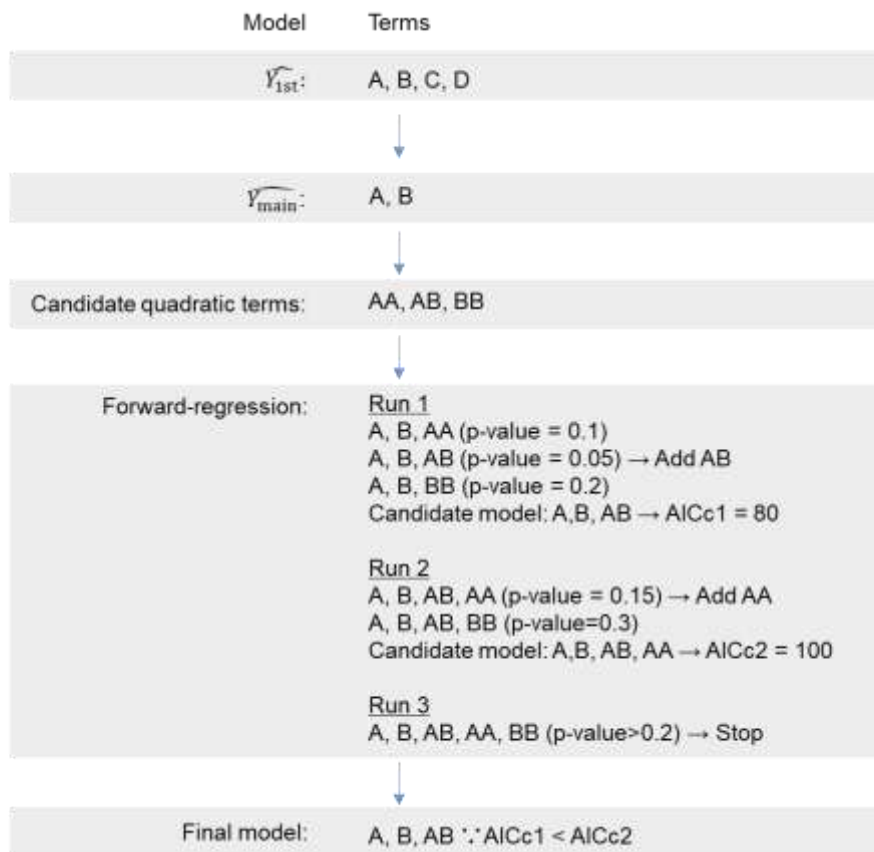


Figure 1 An example of the model selection procedure. In the example, A, B, C, and D are the initial factors. A and B are selected as “main factors.” Among the candidate terms AA, AB, and BB, only AB is incorporated in the final model. The values of p-value and AICc are assumed just for explanation.