MATEMATIKA, 2013, Volume 29, Number 1b, 95-105
©Department of Mathematical Sciences, UTM

An Almost General Code in R to Find Optimal Designs

[1]Ehsan Masoudi, [2]Majid Sarmad and [3]Hooshang Talebi
[1,2]Department of Statistics, Faculty of Mathematical Sciences,
P.O. Box: 91775-1159, Ferdowsi University of Mashhad-Iran
[3]Department of Statistics, Faculty of Sciences
Post Code: 8174673441, University of Isfahan-Iran
e-mail: ehsan.masoudi@stu-mail.um.ac.ir, sarmad@um.ac.ir , h-talebi@sci.ui.ac.ir

Abstract Optimal designs play an important role in many applied areas such as medical and industrial research. Depending the model, finding the optimal design needs an optimization process based on the Fisher information matrix. Using a nice technique in R (a very popular statistical software), an almost general code can be used for many various cases of optimization processes.

Keywords Experimental designs; Locally D-optimal designs; Fisher information matrix; R.

2010 Mathematics Subject Classiffication 62K05

1 Introduction

In many experimental situations, regression techniques are applied for modelling a response of interest in terms of explanatory variables. For example, in biology, Michaelis-Menten model is used to describe the effect of substrate concentration on velocity of the enzyme reaction [1]. In these cases, the models are known and estimating their unknown parameters is desired. The precision of estimates depends on the choice of design in an experiment. The aim of an optimal design problem is to find appropriate design that leads to efficient estimates of parameters in final fitted regression model. Therefore, an exact optimal design is of optimally selecting the number of distinct levels, the explanatory variable levels and weight of each levels with respect to a given optimality criterion. The obtained optimal design can be verified by equivalence theorem [2].

Determination of optimal designs can be done analytically or numerically. Then, a computational software is required even for some analytical methods. A web based designed program at http://optimal-design.biostat.ucla.edu/optimal/ applies Matlab to find optimal designs. In addition, Poursina [3] and Roshan-Nezhad [4] addressed optimal design problems in Logistic and Poisson models by using Maple which can bear testimony to the fact that R [5] as a popular statistical programming language[1] is not common to be used for this type of calculations and there are no available packages for finding optimal designs in nonlinear models.

Fortunately, R has powerful functions for manipulating character type variables and can be used to build Information matrix. In this paper, an intelligent technique is nicely applied to automatically construct Fisher information matrix for nonlinear models. In section 2, a short brief on optimal design problem is presented and the mentioned technique is described in section 3.

2 A Short Brief on Optimal Designs

Consider a nonlinear model (a model that is not linear in terms of parameters) with one explanatory variable as

$$y = (x, \boldsymbol{\theta}) + \epsilon \tag{1}$$

where y denotes the dependent variable, $\eta(x, \boldsymbol{\theta})$ is a nonlinear function of the explanatory variable x with a vector of unknown parameters $\boldsymbol{\theta}$ and random errors, ϵ, are assumed to be i.i.d. with zero mean and unknown constant variance. A design is a collection of points of the explanatory variable. As there are ties in this collection with n different support points, it can be written as a weighted form

$$\xi_n = \begin{Bmatrix} x_1, x_2, \cdots, x_n \\ w_1, w_2, \cdots, w_n \end{Bmatrix}, \quad \sum_{i=1}^{n} w_i = 1, \qquad i = 1, \cdots, n.$$

In general, a unique optimal design does not exist and using different criteria ends with various designs. The criteria depend on purpose of experimenter. If the aim is to estimate the parameters in the model, reasonable one is determinant of Fisher information matrix and the design ξ is D-optimal if maximizes this determinant. It should be noted that the number of support points must be at least as equal as the number of parameters to avoid
singularity of the matrix. When the function $\eta(x, \boldsymbol{\theta})$ in (1) is differentiable with continuous derivative for all elements of $\boldsymbol{\theta}$, the Fisher information matrix for a design ξ will be

$$M(\xi, \boldsymbol{\theta}) = \sum_{i=1}^{n} w_i m(x_i, \boldsymbol{\theta}), \tag{2}$$

where

$$m(x_i, \boldsymbol{\theta}) = \begin{bmatrix} \left(\frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_1}\right)^2 & \frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_1} \frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_2} & \cdots & \frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_1} \frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_p} \\ \frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_2} \frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_1} & \left(\frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_2}\right)^2 & \cdots & \frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_2} \frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_p} \frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_1} & \frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_p} \frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_2} & \cdots & \left(\frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_p}\right)^2 \end{bmatrix}.$$

It is obvious that nonlinearity of $\eta(x, \boldsymbol{\theta})$ causes the dependency on unknown parameters in Fisher information matrix which results in a paradox. Chernoff [6] proposed choosing initial guess values for the unknown parameters. This approach leads to a locally optimal design.

Michaelis-Menten model, as following, is an example of nonlinear models

$$E(y) = \frac{\theta_1 x}{\theta_2 + x}. \tag{3}$$

In the next section, a technique is described to automatically construct the Fisher information matrix for almost nonlinear functions, η, via the Michaelis-Menten model as an example. In addition, the performance of this algorithm is evaluated by Exponantial and Log-linear models as two another nonlinear models.

3 Auto-Constructing the Fisher Information Matrix Determinant by R

To obtain D-optimal design, it is necessary to define (2) for a given model. Unfortunately, R is limited to handle the symbolic computation (D is an example of it), but it has a powerful functions that can work with string and character variables such as gsub, paste and eval which play an important role in the technique [7].

Generally, auto-constructing technique for Fisher information matrix in (2) can be described as the following algorithm:

Step 1 Set input variables:

- npar: Number of parameters in model
- n: Number of support points (by default is set to npar)
- ymean: $\eta(x_i, \boldsymbol{\theta})$ as a character string
- vp: $\boldsymbol{\theta}^0$, vector of initial values for the parameters in locally optimal design approach.

Step 2 For each *i*, construct matrix mc_i, $m(x_i, \boldsymbol{\theta})$ as string.

Step 3 Construct matrix Mc, $M(\xi, \boldsymbol{\theta})$, as string.

Step 4 Substitute parameter symbols in Mc with the given initial values of parameter (vp elements).

Step 5 Substitute x_i 's and w_i 's with symbolic elements of a vector which will be used in optimization process.

Step 6 Construct the string definition of a function for Fisher information matrix determinant as character string.

Step 7 Evaluate the string function to make a real function of Information matrix determinant (ready to use for maximization).

By coding the above algorithm within R workspace, the Fisher information matrix for the models of form (1) can be constructed without user-inerfere and just by setting the appropriate input variables. Hence, selecting the suitable input variables in step 1 has a crucial importance in performance of the algorithm. For instance, to construct the two-point design Fisher information matrix for Michaelis-Menten model with $\boldsymbol{\theta}^0 = (\theta_1 = 1.3, \theta_2 = 1)$, the input variables must defined as following:

```
npar=2
n=2
ymean="(theta1*xi)/(theta2+xi)"
vp=c(1.3,1)
```

As can be seen in the of ymean definition, xi and theta1, theta2, ... must be used as notations for θ_2, θ_2, ... and x_i, respectively. Following tables contain a few key pieces of R codes in each step for the above algorithm.

Table 1 Constructing $m(x_i, \boldsymbol{\theta})$ for each *i* (step 2)

ymean=gsub("i",i,ymean)	Substitute the "i" character with real valued of i in ymean
row=paste("theta",k,sep="") charac1=paste("D(expression(",ymean,") ,","' ",row," '",")",sep="")	Construct $\frac{\partial \eta(x_i, \boldsymbol{\theta})}{\partial \theta_k}$ as a character string, where *k* is the number of rows in $m(x_i, \boldsymbol{\theta})$

col=paste("theta",j,sep="")
charac2=paste("D(expression(",ymean,")
,"," ' ",col," ' ", ")",sep="")
	Construct $\frac{\partial \eta(x_i,\boldsymbol{\theta})}{\partial \theta_j}$ as a character string, where j is the number of columns in $m(x_i,\boldsymbol{\theta})$.

a1=eval(parse(text=(charac1)))	Evaluate charac1 to compute $\frac{\partial \eta(x_i,\boldsymbol{\theta})}{\partial \theta_k}$
a2=eval(parse(text=(charac2)))	Evaluate charac2 to compute $\frac{\partial \eta(x_i,\boldsymbol{\theta})}{\partial \theta_j}$

a1=as.character(as.expression(a1))
a2=as.character(as.expression(a2))
	Convert type of a1 and a2 from "language" to "character"

paste(a1,a2,sep ="*")	Paste a1 and a2 to construct $\frac{\partial \eta(x_i,\boldsymbol{\theta})}{\partial \theta_k}$ $\frac{\partial \eta(x_i,\boldsymbol{\theta})}{\partial \theta_j}$

The constructed mc_i (i = 1) for Michaelis-Menten model is:

```
     [,1]
[1,] "x1/(theta2 + x1)*x1/(theta2 + x1)"
[2,] "x1/(theta2 + x1)*-((theta1 * x1)/(theta2 + x1)^2)"
     [,2]
[1,] "- ((theta1 * x1)/(theta2 + x1)^2)*x1/(theta2 + x1)"
[2,] "- ((theta1 * x1)/(theta2 + x1)^2)*-((theta1 * x1)/(theta2 + x1)^2)"
```

In step three, the Mc should initially be set as an empty string by Mc = " ". The other stages of step three is described in table (2) and will be looped for each i.

Table 2 Constructing matrix $M(\xi,\boldsymbol{\theta})$ (step 3)

weight=paste("w",i,sep="")
mcw=paste(weight,mc_i,sep="*")
	Construct $w_i m(x_i,\boldsymbol{\theta})$

Mc=paste(Mc,mcw,sep="+")	Stepwise summation of mcw

Table 3 Substitute parameter symbols in Mc with given initial values (step 4)

par_symbol=paste("theta",j,sep="")
Mc=gsub(par_symbol,vp[j],Mc)
	Substitute the j-th parameter symbol with vp[j], the jth element of $\boldsymbol{\theta}^0$

Now, Mc for Michaelis-Menten model is

```
[1] "+w1*x1/(theta2 + x1)*x1/(theta2 + x1)
    +w2*x2/(theta2 + x2)*x2/(theta2 + x2)"
[2] "+w1*x1/(theta2 + x1)*-((theta1 * x1)/(theta2 + x1)^2)
    +w2*x2/(theta2 + x2)*-((theta1 * x2)/(theta2 + x2)^2)"
[3] "+w1*-((theta1 * x1)/(theta2 + x1)^2)*x1/(theta2 + x1)
    +w2*-((theta1 * x2)/(theta2 + x2)^2)*x2/(theta2 + x2)"
```

```
[4] "+w1*-((theta1 * x1)/(theta2 + x1)^2)*-((theta1 * x1)/(theta2 + x1)^2)
     +w2*-((theta1 * x2)/(theta2 + x2)^2)*-((theta1 * x2)/(theta2 + x2)^2)"
```

.

In step five, x_i 's and w_i 's will be substituted with $q[.]$, respectively, to achieve the standards of being an objective function in R optimization functions arguments. It is also be applicable any extra information from theorems on the designs of some nonlinear models, such as equality of weights or even adding penalty function as a character string [8].

Table 4 Constructing the string definition of a function for Fisher information matrix determinant (step 6)

Mc=paste(Mc[1:((npar)*(npar))], sep=',',collapse=",")	Concatenate all elements of MC matrix into a single string
charac<-paste("detM=function(q) {Mat<-matrix(c(", Mc,"),npar,npar ;d=det(Mat);return(d)}",sep="")	Paste the required codes to Mc string

In step seven, eval(parse(text=charac)) defines the function of detM in the global environment:

```
function(q){Mat<-matrix(c(
+q[3] * 1/(1) * q[1]/(1 + q[1]) * q[1]/(1 + q[1])
+q[4] * 1/(1) * q[2]/(1 + q[2]) * q[2]/(1 + q[2]),
+q[3] * 1/(1) * -((1.3 * q[1])/(1 + q[1])^2) * q[1]/(1 + q[1])
+q[4] * 1/(1) * -((1.3 * q[2])/(1 + q[2])^2) * q[2]/(1 + q[2]),
+q[3] * 1/(1) * q[1]/(1 + q[1]) * -((1.3 * q[1])/(1 + q[1])^2)
+q[4] * 1/(1) * q[2]/(1 + q[2]) * -((1.3 * q[2])/(1 + q[2])^2),
+q[3] * 1/(1) * -((1.3 * q[1])/(1 + q[1])^2) * -((1.3 * q[1])/(1 + q[1])^2)
+ q[4] * 1/(1) * -((1.3 * q[2])/(1 + q[2])^2) * -((1.3 * q[2])/(1 + q[2])^2
)),2,2);d=det(Mat);return(d)}
```

3.1 Performance Evaluation of auto-constructing Algorithm

Exponential and Log-linear models as two another examples of nonlinear dose response models can be written as (1), where $\eta(x,\boldsymbol{\theta}) = \theta_1 + \theta_2 \exp(x/\theta_3)$ for Exponential and $\eta(x,\boldsymbol{\theta}) = \theta_1 + \theta_2 \log(x+\theta_3)$ for Log-linear model. Because of the broad applicability of these models, efficient experimental designs are important [9]. To construct Fisher informtion matrix of three-points design at $\boldsymbol{\theta}^0$ = (1.6, 1.3, 1) the input variables must be npar = 3, n=3, vpar=c(1.6,1.3,1) with ymean=theta1+theta2*exp(xi/theta3) and ymean=theta1+theta2*log(xi+theta3) for Exponential and Log-linear models, respectively. Then, by applying auto-constructing algorithm the constructed information matrix for Exponential model will be

```
function(q){Mat<-matrix(c(
+q[4] * 1/(1) * 1 * 1 + q[5] * 1/(1) * 1 * 1
+q[6] * 1/(1) * 1 * 1, +q[4] * 1/(1) * exp(q[1]/1) * 1
+q[5] * 1/(1) * exp(q[2]/1) * 1 + q[6] * 1/(1) * exp(q[3]/1) * 1,
+q[4] * 1/(1) * -(1.3 * (exp(q[1]/1) * (q[1]/1^2))) * 1
+q[5] * 1/(1) * -(1.3 * (exp(q[2]/1) * (q[2]/1^2))) * 1
+q[6] * 1/(1) * -(1.3 * (exp(q[3]/1) * (q[3]/1^2))) * 1,
+q[4] * 1/(1) * 1 * exp(q[1]/1) + q[5] * 1/(1) * 1 * exp(q[2]/1)
+q[6] * 1/(1) * 1 * exp(q[3]/1),
+q[4] * 1/(1) * exp(q[1]/1) * exp(q[1]/1)
+q[5] * 1/(1) * exp(q[2]/1) * exp(q[2]/1)
+q[6] * 1/(1) * exp(q[3]/1) * exp(q[3]/1),
+q[4] * 1/(1) * -(1.3 * (exp(q[1]/1) * (q[1]/1^2))) * exp(q[1]/1)
+q[5] * 1/(1) * -(1.3 * (exp(q[2]/1) * (q[2]/1^2))) * exp(q[2]/1)
+q[6] * 1/(1) * -(1.3 * (exp(q[3]/1) * (q[3]/1^2))) * exp(q[3]/1),
+q[4] * 1/(1) * 1 * -(1.3 * (exp(q[1]/1) * (q[1]/1^2)))
+q[5] * 1/(1) * 1 * -(1.3 * (exp(q[2]/1) * (q[2]/1^2)))
+q[6] * 1/(1) * 1 * -(1.3 * (exp(q[3]/1) * (q[3]/1^2))),
+q[4] * 1/(1) * exp(q[1]/1) * -(1.3 * (exp(q[1]/1) * (q[1]/1^2)))
+q[5] * 1/(1) * exp(q[2]/1) * -(1.3 * (exp(q[2]/1) * (q[2]/1^2)))
+q[6] * 1/(1) * exp(q[3]/1) * -(1.3 * (exp(q[3]/1) * (q[3]/1^2))),
+q[4] * 1/(1) * -(1.3 * (exp(q[1]/1) * (q[1]/1^2))) * -(1.3 *
(exp(q[1]/1) *(q[1]/1^2)))
+q[5] * 1/(1) * -(1.3 * (exp(q[2]/1) * (q[2]/1^2))) * -(1.3 *
(exp(q[2]/1) *(q[2]/1^2)))
+q[6] * 1/(1) * -(1.3 * (exp(q[3]/1) * (q[3]/1^2))) * -(1.3 *
(exp(q[3]/1)*(q[3]/1^2)))),
3,3);d=det(Mat);return(d)}
```

while this matrix for Log-linear model is constructed as following R function

```
function(q){Mat<-matrix(c(
+q[4] * 1/(1) * 1 * 1 + q[5] * 1/(1) * 1 * 1
+q[6] * 1/(1) * 1 * 1, +q[4] * 1/(1) * log(q[1] + 1) * 1
+q[5] * 1/(1) * log(q[2] + 1) * 1 + q[6] * 1/(1) * log(q[3] + 1) * 1,
+q[4] * 1/(1) * 1.3 * (1/(q[1] + 1)) * 1
+q[5] * 1/(1) * 1.3 * (1/(q[2] + 1)) * 1
+q[6] * 1/(1) * 1.3 * (1/(q[3] + 1)) * 1,
+q[4] * 1/(1) * 1 * log(q[1] + 1)
+q[5] * 1/(1) * 1 * log(q[2] + 1)
+q[6] * 1/(1) * 1 * log(q[3] + 1),
+q[4] * 1/(1) * log(q[1] + 1) * log(q[1] + 1)
+q[5] * 1/(1) * log(q[2] + 1) * log(q[2] + 1)
+q[6] * 1/(1) * log(q[3] + 1) * log(q[3] + 1),
+q[4] * 1/(1) * 1.3 * (1/(q[1] + 1)) * log(q[1] + 1)
+q[5] *1/(1) * 1.3 * (1/(q[2] + 1)) * log(q[2] + 1)
+q[6] * 1/(1) * 1.3 * (1/(q[3] + 1)) * log(q[3] + 1),
+q[4] * 1/(1) * 1 * 1.3 * (1/(q[1] + 1))
+q[5] * 1/(1) * 1 * 1.3 * (1/(q[2] + 1))
+q[6] * 1/(1) * 1 * 1.3 * (1/(q[3] + 1)),
+q[4] * 1/(1) * log(q[1] + 1) * 1.3 * (1/(q[1] + 1))
+q[5] * 1/(1) * log(q[2] + 1) * 1.3 * (1/(q[2] + 1))
+q[6] * 1/(1) * log(q[3] + 1) * 1.3 * (1/(q[3] + 1)),
+q[4] * 1/(1) * 1.3 * (1/(q[1] + 1)) * 1.3 * (1/(q[1] + 1))
+q[5] * 1/(1) * 1.3 * (1/(q[2] + 1)) * 1.3 * (1/(q[2] + 1))
+q[6] * 1/(1) * 1.3 * (1/(q[3] + 1)) * 1.3 * (1/(q[3] + 1))),
3,3);d=det(Mat);return(d)}
```

4 Discussion and Results

In spite of the fact that R has many powerful packages for optimization2, the most of them are unable to deal with D-optimal problems in nonlinear models. In fact, changing the values of support points leads to a large value (NaN) for the determinant and optimization will be terminated with no result. It seems that the best available package for our purpose is Rsolnp that does a general nonlinear optimization using augmented Lagrange multiplier method [10]. In the current work, gosolnp function from Rsolnp is applied which is random initialization and multiple restarts of the solnp solver and also can carry out a constraint optimization that is required in an optimal design problem because of $\sum_{i=1}^{n} w_i = 1$.

As a result, numerical local D-optimal designs for Michaelis-Menten, Exponential and Log-linear models are given in tables (5), (6) and (7), respectively, with design space [0, 10]. The obtained designs in these tables is exactly as same as the analytical results that are given in [1] and [9]. The other part of coding (in R) has been done, by the authors, to verify the optimality of these designs based on the equivalence theorem and needs to be documented. As can be seen from these tables, the local D-optimal designs are robust with respect to θ_1 for Michaelis-Menten model and with respect to θ_1 and θ_2 for Expnential and Log-linear models. Hence, the local D-optimal designs for these models are relatively robust with respect to misspecication of the model parameters.

Moreover, local D-optimal designs for Weibull, Richards and Inverse Quadratic can also be easily obtained by setting the appropriate input variables.

Table 5 Some locall D-optimal designs in the Michaelis-Menten model with the design space $0 \le x \le 10$

	θ_1^0	θ_2^0	x_1	x_2	w_1
w_2					
1.3	1.0	0.833	10	0.5	0.5
1.6	1.0	0.833	10	0.5	0.5
1.9	1.0	0.833	10	0.5	0.5
2.2	1.0	0.833	10	0.5	0.5
2.5	1.0	0.833	10	0.5	0.5
2.8	1.0	0.833	10	0.5	0.5
3.1	1.0	0.833	10	0.5	0.5
3.4	1.0	0.833	10	0.5	0.5
3.7	1.0	0.833	10	0.5	0.5
4.0	1.0	0.833	10	0.5	0.5
4.3	1.0	0.833	10	0.5	0.5
1.0	1.3	1.032	10	0.5	0.5
1.3	1.3	1.032	10	0.5	0.5
1.6	1.3	1.032	10	0.5	0.5
1.9	1.3	1.032	10	0.5	0.5
2.2	1.3	1.032	10	0.5	0.5
2.5	1.3	1.032	10	0.5	0.5
2.8	1.3	1.032	10	0.5	0.5
3.1	1.3	1.032	10	0.5	0.5
3.4	1.3	1.032	10	0.5	0.5
3.7	1.3	1.032	10	0.5	0.5
4.0	1.3	1.032	10	0.5	0.5
4.3	1.3	1.032	10	0.5	0.5
1.0	1.6	1.212	10	0.5	0.5
1.3	1.6	1.212	10	0.5	0.5
1.6	1.6	1.212	10	0.5	0.5
1.9	1.6	1.212	10	0.5	0.5
2.2	1.6	1.212	10	0.5	0.5
2.5	1.6	1.212	10	0.5	0.5
2.8	1.6	1.212	10	0.5	0.5
3.1	1.6	1.212	10	0.5	0.5
3.4	1.6	1.212	10	0.5	0.5
3.7	1.6	1.212	10	0.5	0.5

4.0	1.6	1.212	10	0.5	0.5
4.3	1.6	1.212	10	0.5	0.5
1.0	1.9	1.377	10	0.5	0.5
1.3	1.9	1.377	10	0.5	0.5
1.6	1.9	1.377	10	0.5	0.5
1.9	1.9	1.377	10	0.5	0.5
2.2	1.9	1.377	10	0.5	0.5
2.5	1.9	1.377	10	0.5	0.5
2.8	1.9	1.377	10	0.5	0.5
3.1	1.9	1.377	10	0.5	0.5
3.4	1.9	1.377	10	0.5	0.5
3.7	1.9	1.377	10	0.5	0.5
4.0	1.9	1.377	10	0.5	0.5
4.3	1.9	1.377	10	0.5	0.5
1.0	2.2	1.528	10	0.5	0.5

Table 6 Some locall D-optimal designs in the Exponential dose response model with the design space $0 \le x \le 10$

θ_1^0	θ_2^0	θ_3^0	x_1	x_2	x_3	w_1	w_2	w_3
1.6	1.3	1.0	0	9.000	10	0.333	0.333	0.333
1.9	1.3	1.0	0	9.000	10	0.333	0.333	0.333
1.0	1.6	1.0	0	9.000	10	0.333	0.333	0.333
1.3	1.6	1.0	0	9.000	10	0.333	0.333	0.333
1.6	1.6	1.0	0	9.000	10	0.333	0.333	0.333
1.9	1.6	1.0	0	9.000	10	0.333	0.333	0.333
1.0	1.9	1.0	0	9.000	10	0.333	0.333	0.333
1.3	1.9	1.0	0	9.000	10	0.333	0.333	0.333
1.6	1.9	1.0	0	9.000	10	0.333	0.333	0.333
1.9	1.9	1.0	0	9.000	10	0.333	0.333	0.333
1.0	1.0	1.3	0	8.705	10	0.333	0.333	0.333
1.3	1.0	1.3	0	8.705	10	0.333	0.333	0.333
1.6	1.0	1.3	0	8.705	10	0.333	0.333	0.333
1.9	1.0	1.3	0	8.705	10	0.333	0.333	0.333
1.0	1.3	1.3	0	8.705	10	0.333	0.333	0.333
1.3	1.3	1.3	0	8.705	10	0.333	0.333	0.333
1.6	1.3	1.3	0	8.705	10	0.333	0.333	0.333
1.9	1.3	1.3	0	8.705	10	0.333	0.333	0.333
1.0	1.6	1.3	0	8.705	10	0.333	0.333	0.333
1.3	1.6	1.3	0	8.705	10	0.333	0.333	0.333
1.6	1.6	1.3	0	8.705	10	0.333	0.333	0.333
1.9	1.6	1.3	0	8.705	10	0.333	0.333	0.333
1.0	1.9	1.3	0	8.705	10	0.333	0.333	0.333
1.3	1.9	1.3	0	8.705	10	0.333	0.333	0.333
1.6	1.9	1.3	0	8.705	10	0.333	0.333	0.333
1.9	1.9	1.3	0	8.705	10	0.333	0.333	0.333
1.0	1.0	1.6	0	8.419	10	0.333	0.333	0.333
1.3	1.0	1.6	0	8.419	10	0.333	0.333	0.333
1.6	1.0	1.6	0	8.419	10	0.333	0.333	0.333
1.9	1.0	1.6	0	8.419	10	0.333	0.333	0.333
1.0	1.3	1.6	0	8.419	10	0.333	0.333	0.333

1.3	1.3	1.6	0	8.419	10	0.333	0.333	0.333
1.6	1.3	1.6	0	8.419	10	0.333	0.333	0.333
1.9	1.3	1.6	0	8.419	10	0.333	0.333	0.333
1.0	1.6	1.6	0	8.419	10	0.333	0.333	0.333
1.3	1.6	1.6	0	8.419	10	0.333	0.333	0.333
1.6	1.6	1.6	0	8.419	10	0.333	0.333	0.333
1.9	1.6	1.6	0	8.419	10	0.333	0.333	0.333
1.0	1.9	1.6	0	8.419	10	0.333	0.333	0.333
1.3	1.9	1.6	0	8.419	10	0.333	0.333	0.333
1.6	1.9	1.6	0	8.419	10	0.333	0.333	0.333
1.9	1.9	1.6	0	8.419	10	0.333	0.333	0.333
1.0	1.0	1.9	0	8.152	10	0.333	0.333	0.333
1.3	1.0	1.9	0	8.152	10	0.333	0.333	0.333
1.6	1.0	1.9	0	8.152	10	0.333	0.333	0.333
1.9	1.0	1.9	0	8.152	10	0.333	0.333	0.333
1.0	1.3	1.9	0	8.152	10	0.333	0.333	0.333
1.3	1.3	1.9	0	8.152	10	0.333	0.333	0.333

Table 7 Some locall D-optimal designs in the Log-linear dose response model with the design space $0 \le x \le 10$

θ_1^0	θ_2^0	θ_3^0	x_1	x_2	x_3	w_1	w_2	w_3
1.6	1.3	1.0	0	1.638	10	0.333	0.333	0.333
1.9	1.3	1.0	0	1.638	10	0.333	0.333	0.333
1.0	1.6	1.0	0	1.638	10	0.333	0.333	0.333
1.3	1.6	1.0	0	1.638	10	0.333	0.333	0.333
1.6	1.6	1.0	0	1.638	10	0.333	0.333	0.333
1.9	1.6	1.0	0	1.638	10	0.333	0.333	0.333
1.0	1.9	1.0	0	1.638	10	0.333	0.333	0.333
1.3	1.9	1.0	0	1.638	10	0.333	0.333	0.333
1.6	1.9	1.0	0	1.638	10	0.333	0.333	0.333
1.9	1.9	1.0	0	1.638	10	0.333	0.333	0.333
1.0	1.0	1.3	0	1.877	10	0.333	0.333	0.333
1.3	1.0	1.3	0	1.877	10	0.333	0.333	0.333
1.6	1.0	1.3	0	1.877	10	0.333	0.333	0.333
1.9	1.0	1.3	0	1.877	10	0.333	0.333	0.333
1.0	1.3	1.3	0	1.877	10	0.333	0.333	0.333
1.3	1.3	1.3	0	1.877	10	0.333	0.333	0.333
1.6	1.3	1.3	0	1.877	10	0.333	0.333	0.333
1.9	1.3	1.3	0	1.877	10	0.333	0.333	0.333
1.0	1.6	1.3	0	1.877	10	0.333	0.333	0.333
1.3	1.6	1.3	0	1.877	10	0.333	0.333	0.333
1.6	1.6	1.3	0	1.877	10	0.333	0.333	0.333
1.9	1.6	1.3	0	1.877	10	0.333	0.333	0.333
1.0	1.9	1.3	0	1.877	10	0.333	0.333	0.333
1.3	1.9	1.3	0	1.877	10	0.333	0.333	0.333
1.6	1.9	1.3	0	1.877	10	0.333	0.333	0.333
1.9	1.9	1.3	0	1.877	10	0.333	0.333	0.333
1.0	1.0	1.6	0	2.077	10	0.333	0.333	0.333
1.3	1.0	1.6	0	2.077	10	0.333	0.333	0.333
1.6	1.0	1.6	0	2.077	10	0.333	0.333	0.333
1.9	1.0	1.6	0	2.077	10	0.333	0.333	0.333
1.0	1.3	1.6	0	2.077	10	0.333	0.333	0.333
1.3	1.3	1.6	0	2.077	10	0.333	0.333	0.333

1.6	1.3	1.6	0	2.077	10	0.333	0.333	0.333
1.9	1.3	1.6	0	2.077	10	0.333	0.333	0.333
1.0	1.6	1.6	0	2.077	10	0.333	0.333	0.333
1.3	1.6	1.6	0	2.077	10	0.333	0.333	0.333
1.6	1.6	1.6	0	2.077	10	0.333	0.333	0.333
1.9	1.6	1.6	0	2.077	10	0.333	0.333	0.333
1.0	1.9	1.6	0	2.077	10	0.333	0.333	0.333
1.3	1.9	1.6	0	2.077	10	0.333	0.333	0.333
1.6	1.9	1.6	0	2.077l	10	0.333	0.333	0.333
1.9	1.9	1.6	0	2.077	10	0.333	0.333	0.333
1.0	1.0	1.9	0	2.248	10	0.333	0.333	0.333
1.3	1.0	1.9	0	2.248	10	0.333	0.333	0.333
1.6	1.0	1.9	0	2.248	10	0.333	0.333	0.333
1.9	1.0	1.9	0	2.248	10	0.333	0.333	0.333
1.0	1.3	1.9	0	2.248	10	0.333	0.333	0.333
1.3	1.3	1.9	0	2.248	10	0.333	0.333	0.333

5 Conclusion

For optimal design problems, there has not been an effort to apply R for finding optimal designs in nonlinear models because of two reasons. The first reason is lack of symbolic computation in R which causes codding to be time-consumer and boring. The second reason is connected with the element values of Fisher information matrix which exceeds the double-precision (R precision default) for some points of design space and gives rise to halt in optimization process as NaN is produced in calculations. To address the R symbolic computation issue, auto-constructing technique is introduced to construct Fisher information matrix without user-interference. Furthermore, the auto-constructing technique can facilitates the other optimal design approaches. For instance, in minimax D-optimal design algorithm the number of support points may be changed for some iterations (See [11]) and by this technique the Fisher information matrix can be constructed just by setting the value of n through variables. R users can apply this technique for constructing the complicated formulas instead of symbolic computations in R. What is more, function gosolnp from package Rsolnp can be applied to minimize determinant or any function of Fisher information matrix. In fact, while the other nonlinear optimizer such as optim is failed when NaN appears, gosolnp as a multistart algorithm can ignore the produced NaN and carry out the optimization process without halt.

Acknowledgments

This research was supported by "Ordered and Spatial Data Center of Excellence" of Ferdowsi University of Mashhad.

References

[1] Dette, H., Melas, V.B. and Wong, W. K. Optimal Design for Goodness-of-Fit of the Michaelis-Menten Enzyme Kinetic Function. *Journal of the American Statistical Association*. 2005. 100(472): 1370-1381.

[2] Kiefer, J. C. General equivalence theory for optimum designs (approximate theory). *Ann. Statist.* 1974. 2: 849-879.

[3] Poursina, D. *Design of experiments for generalized linear models*. University of Isfahan: Master Thesis. 2007.

[4] Roshan-Nezhad, M. *Optimal experimental designs for the Poisson regression model*. Ferdowsi University of Mashhad: Master Thesis. 2009.

[5] R Development Core Team, 2011, *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, http://www.R-project.org/.

[6] Chernou_, H. Locally optimal designs for estimating parameters. Ann. Math. Statist. 1953. 24: 586-602.

[7] Crawley, M. J. *The R book*. Chichester, UK: John Wiley & Sons. 2007.

[8] Masoudi, E. *Finding locally D-optimal designs in R (An almost general code applicable in most models)*. Ferdowsi University of Mashhad: Master Thesis. 2012.

[9] Dette, H., Kiss, C., Bevanda, M. and Bretz, F. Optimal designs for the EMAX, log-linear and exponential model. *Biometrika*. 2010. 97(2): 513-518.

[10] Ghalanos, A. and Theussl, S. *Rsolnp: General nonlinear optimization using augmented Lagrange multiplier method*. R package: version 1.11. 2011.

[11] King, J. and Wong, W. K. Minimax D-Optimal Designs for the Logistic Model. *Biometrics*. 2000. 56: 1263-1267.