Design of Dynamic Experiments to Optimize Batch Processes

A Senior Thesis for the Department of Chemical & Biological Engineering

William Alexander Marvin

Advisor: Professor Christos Georgakis

Examining Committee: Professor Christos Georgakis and Professor Kyongbum Lee

Tufts University, 2009

Table of Contents

1. Background

1.1 Batch Processes

It often requires too much investment to create accurate models for complex batch unit operations and reactors. Small production rates affiliated with batch processes do not offer enough economic benefit for that level of optimization. Simpler optimization techniques that do not require understanding of the process inner workings are more suited to these operations. For this reason the use of Design of Experiments (DoE) and evolutionary optimization are applied to many industrial pharmaceutical applications including: blending, crystallization, tableting, granulation, extrusion-spheronization, drying, and coating.

Many batch processes can be categorized as either batch or fed-batch reactors. Optimization of these processes involves determining the best inputs (factors) to maximize an output (response). Process factors may include reactor temperature, pressure, batch time, initial concentrations, feed flow rate, and reactor volume. The response variable could be the yield, selectivity, particle size, hardness, film thickness, productivity, profit or a weighted combination of these.

Certainly in any given batch process there is a nearly infinite number of factors that may influence the response, and some thought and preliminary experimentation may be required to determine the most important ones. For the purposes of examining the proposed methodology, it is assumed that a small number of important factors have been identified as having the largest influence to the process.

1.2 Design of Experiments (DoE)

DoE allows for a process to be optimized quickly with only a few planned experiments without knowledge of the underlying physical or chemical phenomena. A statistical approach is applied in planning the experiments in order to draw meaningful conclusions from the results. Factorial experimentation is done in which factors are varied together in order to understand how the factors interact. Interaction between factors cannot be determined if a simpler “one-factor-at-a-time” experimental design is used.

1.3 DoE Design Space

Deciding on how to vary the factors, and thus what experiments to run, is a question of what design space to use. The experimenter has to decide what ranges of the factors to test and at how many different values (levels). If, for example, the impact of a factor, A, on the response wants to be understood over the range $-1 \leq A \leq 1$, the experimenter might choose to test values of A = -1, 0, and 1 in a 3-level design. A 3-level design will give more information than a 2-level design, allowing for a quadratic relationship between the value of A and the value of the response variable, at the cost of performing more experiments.

Factorial experimental designs involve two or more factors, each examined at two or more levels. These designs, introduced by R.A. Fischer in 1935, are statistically powerful because they are symmetric, and thus easy to analyze for effects and interactions between factors.[1] Analysis of Variance (ANOVA) can be performed easily to analyze the statistical significance of a factor's interaction on the response.[2] For visualization, Figure 1 shows the graphed design space for some full factorial designs analyzing 3 factors. For 3 factors, a factorial design with 2 levels requires 8 ($= 3^2$) experiments and with 3 levels requires 27 experiments ($= 3^3$).

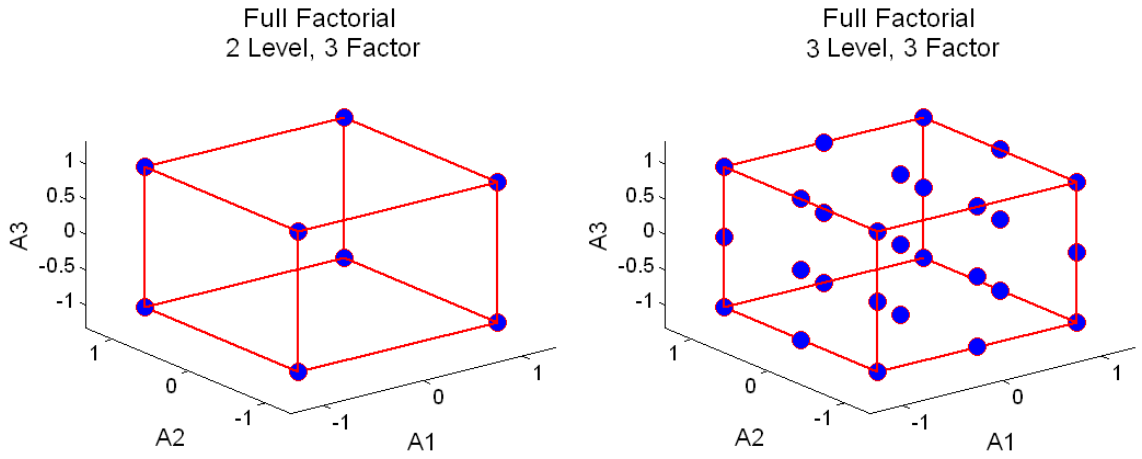

Figure 1. Full Factorial Design Spaces for 3 Factors, 2-Level (Left) and 3-Level (Right)

Performing large numbers of experiments can be avoided when the experiments are time consuming or expensive by using other designs. Central composite designs are used to slightly expand a 2-level factorial design so that quadratic relationships can be found between the factors and the response. This is accomplished by only increasing the number of experiments required by 7 when compared to the 2-level factorial design. A few of these designs for the 3-factor case

are shown in Figure 2. Generally, the center point of the design is replicated, which gives more accuracy around the center of the design space.

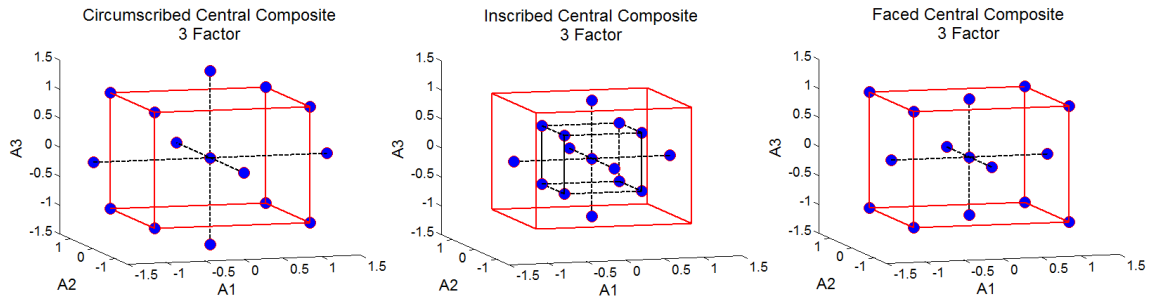

Figure 2. Central Composite Design Spaces for 3 Factors (Circumscribed, Central, and Faced)

The designs discussed previously have all been complete experimental plans laid out before experimentation begins. Designs exist, however, that will use information from the set of completed experiments to determine what experiments to perform next. Advantages of this type of design would be a reduced number of experiments and more knowledge about the region near the expected optimum. A disadvantage is that the generated model may not be generally applicable to regions far from the calculated optimum. These design methods fall under the category of Optimal Designs. D-Optimal Designs, for example, seek to maximize the information content of the model parameters, making the model more accurate.[3] To maintain generality of the model and simplicity of analysis for testing the proposed new methodology, full factorial designs will be used here.

1.4 Response Surface

Once all of the experiments have been performed, empirical response surfaces can be generated to predict with reasonable accuracy the outcomes of the system. This is done by performing multi-linear regression to determine how the factors are related to the response. Thought should be put into what terms to include into the model. Before any calculation is performed it is reasonable to think that quadratic relationships between a 2-level factor and the response variable cannot be obtained. Interactions (one factor multiplied by another) can be present with 2-level factors though. Table 1 shows the terms that are included in 2-level, 3-factor designs along with the additional terms for 3-level, 3-factor designs.

Table 1. Number of Model Terms for 3-Factor Designs (2-Level and 3-Level)

2-Level Terms	3-Level Additional Terms
constant	A_1^2
A_1	A_2^2
A_2	A_3^2
A_3	$A_1^2A_2$
A_1A_2	$A_1^2A_3$
A_1A_3	$A_1A_2^2$
A_2A_3	$A_2^2A_3$
$A_1A_2A_3$	$A_1A_3^2$
	$A_2A_3^2$

The response surface model is a linear combination of these terms and can be written in the following form:

$$y_i = \beta_0 + \left(\sum_{j=1}^{p} \beta_j x_{ij} \right) + \varepsilon_i \ , \quad x_{ij} = f_j\left(A_{1,i}, A_{2,i}, \ldots, A_{k,i}\right) \tag{1}$$

y_i = response of experiment i
p = number of model terms (not including the constant)
k = number of factors
β_j = coefficient of term j
x_{ij} = term j for experiment i and is a function of the inputs
ε_i = error (deviation from model)

Or the following matrix form:

$$\underline{Y} = \underline{\underline{X}}\,\underline{\beta} + \underline{\varepsilon} \tag{2}$$

$$\underline{\underline{X}} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \\ 1 & x_{n1} & \cdots & & x_{np} \end{bmatrix}$$

n = number of experimental runs

It should be noted that the first column in $\underline{\underline{X}}$ corresponds to the constant term in the model and thus is all ones. Obtaining estimates for the model coefficients is shown below:

$$\begin{aligned} \underline{Y} &= \underline{\underline{X}}\,\underline{\hat{\beta}} \\ \underline{\underline{X}}^T\underline{Y} &= \left(\underline{\underline{X}}^T\underline{\underline{X}}\right)\underline{\hat{\beta}} \\ \underline{\hat{\beta}} &= \left(\underline{\underline{X}}^T\underline{\underline{X}}\right)^{-1}\underline{\underline{X}}^T\underline{Y} \end{aligned} \tag{3}$$

In practice rigorous statistical testing is done to determine which model terms have a significant impact on the response variable, and should be included in the response surface model. The key tests used in this paper will be analysis of variance on the model as a whole and t-tests on the individual model coefficients. The null hypothesis for these tests is that a coefficient value is equal to 0. A primary assumption of these tests is that the error is approximately normally distributed with a finite variance and a mean of zero.[4] A brief derivation is shown for the t-test for individual coefficients. Assuming:

$$\varepsilon_i \sim N\left(0,\sigma^2\right) \tag{4}$$

Then an unbiased estimator for σ^2 is: $\hat{\sigma}^2 = \dfrac{S}{n-p}$, where $S = \sum_{i=1}^{n}\varepsilon_i^2$. The relationship follows a Chi-squared distribution with n-p degrees of freedom.

$$\hat{\sigma}^2\frac{n-p}{\sigma^2} \sim \chi^2_{n-p} \tag{5}$$

From Equation 3 it can be shown that the estimates for the model coefficients have the following multivariate normal distribution:[5]

$$\underline{\hat{\beta}} \sim N\left(\underline{\beta}, \sigma^2\left(\underline{\underline{X}}^T\underline{\underline{X}}\right)^{-1}\right) \tag{6}$$

The standard error for the model coefficient can then be found to be:

$$\hat{\sigma}_j = \sqrt{\frac{S}{n-p-1}\left[\left(\underline{\underline{X}}^T \underline{\underline{X}}\right)^{-1}\right]_{jj}} \tag{7}$$

Thus the model coefficient, and term, is not significant if the value of zero is within the following confidence interval:

$$\hat{\beta}_j \pm t_{\frac{\alpha}{2},n-p-1}\hat{\sigma}_j \tag{8}$$

where $t_{\frac{\alpha}{2},n-p-1}$ is a student's t-distribution with n-p-1 degrees of freedom.

Once a final set of model terms with their respective coefficients has been calculated, the response surface can be used for optimization purposes. The optimum inputs are found to either maximize or minimize the response variable, or objective function.

1.5 Optimization

For optimization an objective function, J, is a function of the response variable(s) for which model(s) were created. The objective function is thus a function of the factors (the response surface). Calculation of the optimum values for the factors is also often a constrained problem over an allowable range. In several cases more than one factor may define the boundaries of the allowable range, and thus the constraints on the optimization become more complicated.

To allow for linear and non-linear constraints, the function fmincon in MATLAB® was used for optimization.[6] Gradient information was easily calculated for the model since the terms are all polynomial and passed into the optimization routine. With more complex response surfaces it is possible to have many local minima/maxima, especially at the constraints, which could make optimization to find the global minima/maxima more difficult. To overcome this, multiple initial guesses were used until the global minimum/maximum was found.

1.6 Goal of Design of Dynamic Experiments (DoDE)

DoE requires static inputs (factors) which remain constant throughout the process, but many optimized processes require dynamic inputs. For example: the optimal temperature profile for

crystallization is likely dynamic rather than static at a set point. Similarly, it may be the case where reactor pressure or feed rate is a dynamic variable. With existing controllers these dynamic variables can be quite easy to control, but the methodology is not available for design of dynamic experiments (DoDE). DoDE methodology is designed for optimizing these dynamic process variables without knowledge of the underlying kinetics, much like DoE has been used for with respect to static process variables.

2. Design of Dynamic Experiments (DoDE)

2.1 Dimensionless Variables

A dynamically changing input must be described in terms of a finite set of factors to allow for design of experiment methodology to be applied. Additionally, it is useful to make all process variables dimensionless. Process reaction time is made dimensionless as $\tau = t/t_b$, where t_b is the batch time, and thus spans [0, 1]. All factors are also scaled, where possible, to range from -1 to 1.

2.2 Dynamic Variables

The dynamically changing input is described by the decision variable function, u(τ), considered to be a member of the Hilbert space $\mathcal{L}_2(0,1)$ of square-integrable vector functions. A set of basic-function in that space is denoted as $\{\varphi_i(\tau); i = 1, 2, 3, ...\}$. It has been shown that the decision variable can be written as:[7]

$$\boldsymbol{u}(\tau) = \boldsymbol{u_0} + \Delta \boldsymbol{U}\left(\sum_{i=1}^{\infty} \boldsymbol{a}_i \varphi(\tau)\right) \cong \boldsymbol{u_0} + \Delta U\left(\sum_{i=1}^{N} \boldsymbol{a}_i \varphi(\tau)\right) \tag{9}$$

$$u_0 = \left(u^{\max} + u^{\min}\right)/2$$

$$\Delta \boldsymbol{U} = diag\left(\left(u_i^{\max} - u_i^{\min}\right)/2\right)$$

These expansion coefficients, a_i, are the only unknowns and will be the factors manipulated in this methodology. A small number, *N*, of these terms is required to adequately optimize a

dynamic input. The objective function $J(a_1, a_2, ...)$ is then defined as in Equation 2 with the x_{ij} as a function of the a_i.

Scaling of the decision variable function, u(τ), to stay within the allowed range for the input will be shown later to introduce linear and non-linear constraints on the values of the a_i factors. This decision variable is generally created to have a range of [-1, 1], thus making it useful in the dimensionless applications to these reactor problems.

We selected the shifted Legendre polynomials which is an orthogonal basis set in the Hilbert space $\mathcal{L}_2(0,1)$. They are normalized so that their value at 0 is equal to 1, $\varphi_i(0)=1$. The first seven Legendre polynomials are shown below and the first four are plotted in Figure 3. These orthogonal functions are used to define the decision variable in all the following case studies.

$$\begin{aligned}
\varphi_1(\tau) &= 1 \\
\varphi_2(\tau) &= 1-2\tau \\
\varphi_3(\tau) &= 1-6\tau+6\tau^2 \\
\varphi_4(\tau) &= 1-12\tau+30\tau^2-20\tau^3 \\
\varphi_5(\tau) &= 1-20\tau+90\tau^2-140\tau^3+70\tau^4 \\
\varphi_6(\tau) &= 1-30\tau+210\tau^2-560\tau^3+630\tau^4-252\tau^5 \\
\varphi_7(\varphi) &= 1-42\tau+420\tau^2-1680\tau^3+3150\tau^4-2772\tau^5+924\tau^6
\end{aligned} \tag{10}$$

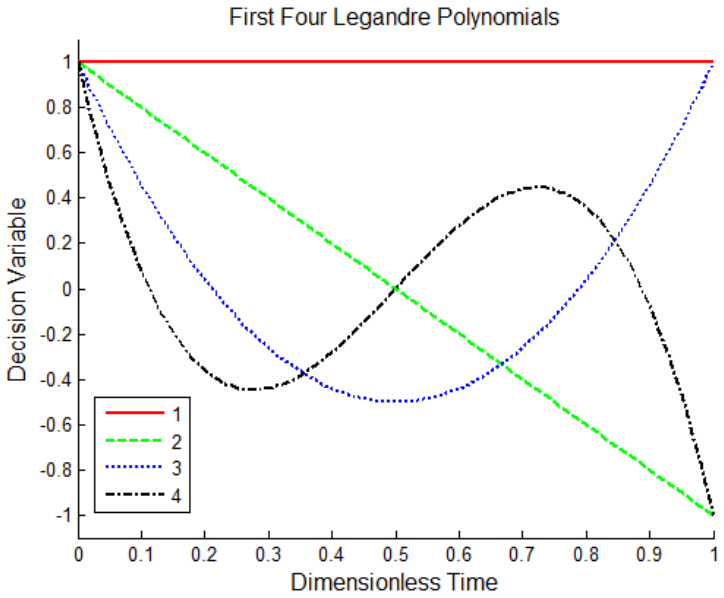

Figure 3. First Seven Legandre Polynomials

2.3 Simulated Experiments

Testing of this methodology required many experiments to be performed. Processes in which all reaction kinetics were known were examined so that the DoDE method could be compared with DoE and exactly solved solutions. This allowed for the experiments to be simulated using software packages like MATLAB®, greatly reducing the time requirement. Other members of Professor Georgakis' research group applied the methodology to actual experiments including crystallization and hydrogen reaction.

The processes examined had single or multiple reactions that could be explained by a small number of non-linear ordinary differential equations (ODEs) as in the following equation:

$$\frac{d\underline{x}}{dt} = \underline{f}\left(\underline{x}, \underline{p}, \underline{u}, t\right); \text{ with } \underline{x}(t=0) = \underline{x}_0 \tag{11}$$

where the parameters $\underline{p}$ are known and the dynamic decision variable is bounded by:
$u_{\min} \le \underline{u}(t) \le u_{\max}$; for $\forall\ t \in [0, t_b]$

Solving the system of ODEs was done using the ode45 function in MATLAB® which utilizes the explicit Runge-Kutta (4,5) formula, the Dormand-Prince pair.[6] The ODE systems were small and non-stiff so this method was sufficient to simulate all experiments.

Experimental error was not inherently present in the simulated experiments. It was added in a fractional manner to the "measured" variables to determine if it had an impact on the performance of the methodology. The simulated experimental responses were stored as a vector, $\underline{PI}$.

$$\begin{aligned} &\underline{PI_e} = \underline{PI} * (1 + \alpha * \underline{e}) \\ &e \sim N(0,1) \\ &\alpha \in \{0, 0.01, 0.02, 0.05, 0.1\} \end{aligned} \tag{12}$$

Normally distributed error of 0%, 1%, 2%, 5% and 10% was examined in the following examples. Beyond this point, the DoDE methodology has very little difference from classical DoE methodology, so will be generally accepted and can be readily implemented.

2.4 Response Surface Creation

Response surface models were created for the DoDE data in the same way that they are created for DoE data. The response surface model (RSM) is an empirical interpolative relationship between the dynamic factors, a_i, and any static factors, A_i, and the response, Y. Multi-linear regression is performed as shown in Equation 3 to determine the model coefficients. In the examples presented in this paper, the response is either the conversion of reactant or the productivity of reactor, but any other pertinent response variable may be examined.

2.5 Statistics

Before optimization was performed, the model was analyzed for statistical significance. As discussed in section 1.4, model terms that were not statistically significant were removed. A stepwise multi-linear regression routine was developed that removed the *least* significant model term by using equation 8 to calculate each coefficient's p-value, and then recalculated the model. This process was repeated until all of the model terms were statistically significant to the $\alpha = 0.05$ level, meaning for each term there was less than 5% chance that its coefficient value was caused only by random noise.

Comparisons were made to determine if this was completely necessary to remove insignificant terms from the model before optimization was performed. The data will be presented later, but in most cases there seemed to be little change in the results. A slight improvement in the variance of the calculated optimums was observed.

Confidence intervals for the model prediction were calculated to determine if the experimental results were within the accuracy of the response surface model. Most importantly, it was hoped that the actual results of an experiment performed with the optimum inputs would compare well with the response surface model prediction.

From error propagation theory, the $100(1-\alpha)\%$ predicted response confidence intervals for the data are given by:[5]

$$\underline{Z}^T \underline{\hat{\beta}} \pm t_{\frac{\alpha}{2},n-p-1} \hat{\sigma} \sqrt{1 + \underline{Z}^T \left(\underline{\underline{X}}^T \underline{\underline{X}} \right)^{-1} \underline{Z}} \tag{13}$$

$\underline{Z}$ is a column vector of the model terms (length = 1+p) evaluated at an input vector, $\underline{x} = [a_1, a_2, \ldots, a_k, A_1, A_2, \ldots, A_w]$, for a process with k dynamic factors and w static factors. If this confidence interval is wide, the response surface model is not very accurate at predicting the optimum point of operation.

2.6 Optimization

Optimization is performed as discussed for classical DoE. Constrained non-linear optimization of the response surface is performed using fmincon in MATLAB® to minimize an objective function, J. For case study 1 and 2, this objective function is the conversion of the initial reactant. For case study 3, the objective function is the process productivity, calculated as the amount of desired substance produced per time. This objective function is closely related to profitability of the batch process, which is especially important in industrial processes.

Multi-start optimization is done using multiple initial guesses and comparing the resulting optima to avoid identifying a local rather than a global minimum. In most cases the corners of the design space along with its center are used as the initial guesses. Only the best of the minima is returned as the global minimum, $\underline{x}_{opt}$.

2.7 Analysis

The confidence interval at $\underline{x}_{opt}$ is calculated to determine the accuracy of the response surface model prediction in that region. An additional simulated experiment is performed using $\underline{x}_{opt}$ as the process inputs to determine the actual response of the simulated nonlinear process. Both the model prediction and the actual response are compared with the best response from the performed experiment. Ideally, the response at $\underline{x}_{opt}$ would be better than the best response from experiment. This was generally found to be true.

3. Case Study 1: 1st Order Reversible Reaction

3.1 Problem Definition

A simple batch reactor was examined in which a reversible reaction between reactant A and product B takes place. Initially the reactor has a reactant concentration of 1M. The kinetics of the reaction is as follows:

$$
\begin{aligned}
& A \underset{k_2}{\overset{k_1}{\rightleftarrows}} B \\
& r = k_1 C_A - k_2 C_B \\
& k_1 = k_{1,0} e^{-E_1/(RT)} \left[=\right] \mathrm{hr}^{-1}, \quad k_2 = k_{2,0} e^{-E_2/(RT)} \left[=\right] \mathrm{hr}^{-1} \\
& k_{1,0} = 1.32 \cdot 10^{7}\ \mathrm{hr}^{-1}, \quad k_{2,0} = 5.24 \cdot 10^{13}\ \mathrm{hr}^{-1} \\
& E_1 = 10{,}000 \frac{cal}{gmol}, \quad E_2 = 20{,}000 \frac{cal}{gmol}
\end{aligned}
\tag{14}
$$

The controlled variable, $u(t)$, for this case study is the reactor temperature. Since the reverse activation energy is greater than the forward, the ideal temperature profile is expected to be decreasing to produce the largest amount of product B.[8] One can imagine a jacketed batch reactor shown in Figure 4 with sufficient control as to produce any reactor vessel temperature profile with time.

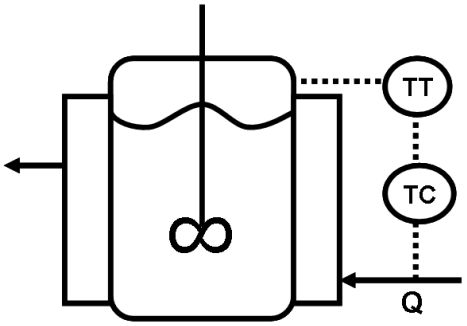

Figure 4. Batch Reactor with Temperature Control

As a simple case study, only dynamic factors are included in this DoDE design. Two or three dynamic factors are used in these designs. Thus only the first three shifted Legendre polynomials

are used for describing the temperature profile, meaning that constant, linear and quadratic polynomials are examined.

3.2 Constraints

The vessel temperature, T, is constrained by a maximum and minimum allowable value. In practice this could be due to physical limitations of the vessel, temperature control system or prior knowledge of the temperature range at which high conversion has been observed. Additionally, the batch time, t_b, is set at 2.5 hours. Process variables can then be made dimensionless as follows:

$$\tau = t/t_b$$
$$u(\tau) = \frac{T(\tau) - T_{ref}}{\Delta T}; \quad T_{ref} = 0.5(T_{\max} + T_{\min}), \Delta T = 0.5(T_{\max} - T_{\min}) \tag{15}$$

For this case study three mean vessel temperatures, T_{ref}, were examined: 35°C, 30°C and 20°C. The allowed temperature range had a width, ΔT = 15°C. Depending on the number of dynamic factors in the design, the design space constraints vary.

3.3 Design Space (2-Factors)

In the 2-factor case only linear temperature profiles are examined. The two dynamic factors, a_1 and a_2, create the following decision variable function:

$$u(\tau) = a_1\varphi_1(\tau) + a_2\varphi_2(\tau) = (a_1 + a_2) - 2a_2\tau; \quad \tau \in [0,1] \tag{16}$$

Due to how $u(\tau)$ was defined in Equation 15, $u(\tau)$ is constrained between -1 and 1. Since the function is linear, only the value at τ = 0 and 1 need to be checked for the whole function to meet this requirement.

$$u(0) = a_1 + a_2, \quad u(1) = a_1 - a_2 \tag{17}$$

Thus there exist the following 4 linear constraints:

$$-1 \le a_1 + a_2 \le 1, \quad -1 \le a_1 - a_2 \le 1 \tag{18}$$

The design space for 2-level and 3-level designs is shown in the following figures. For clarity, graphs are included on the right to demonstrate the temperature profiles (dimensionless) tested for each experiment.

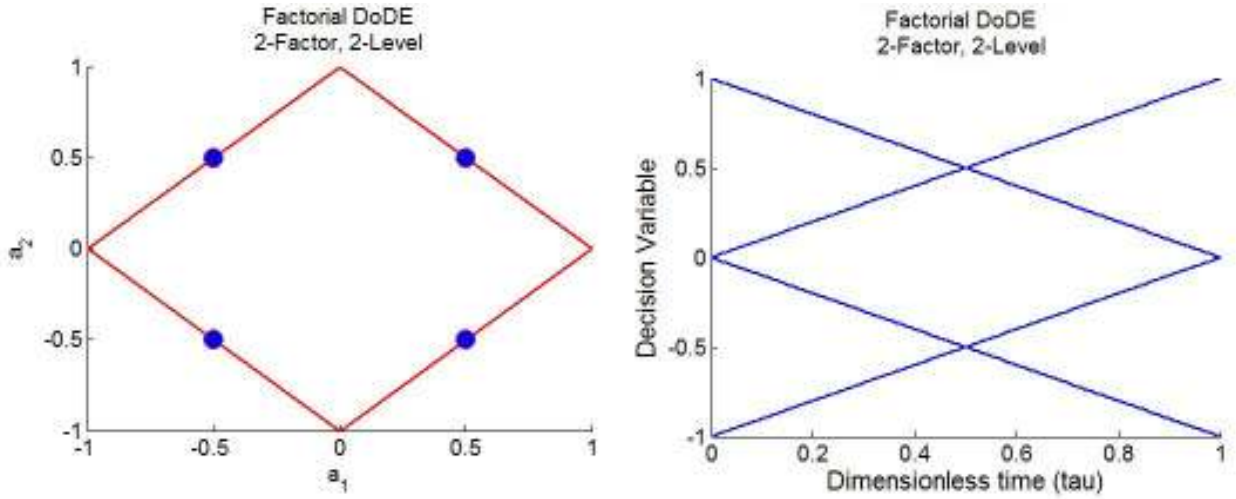

Figure 5. Factorial DoDE for 2-Factor, 2-Level

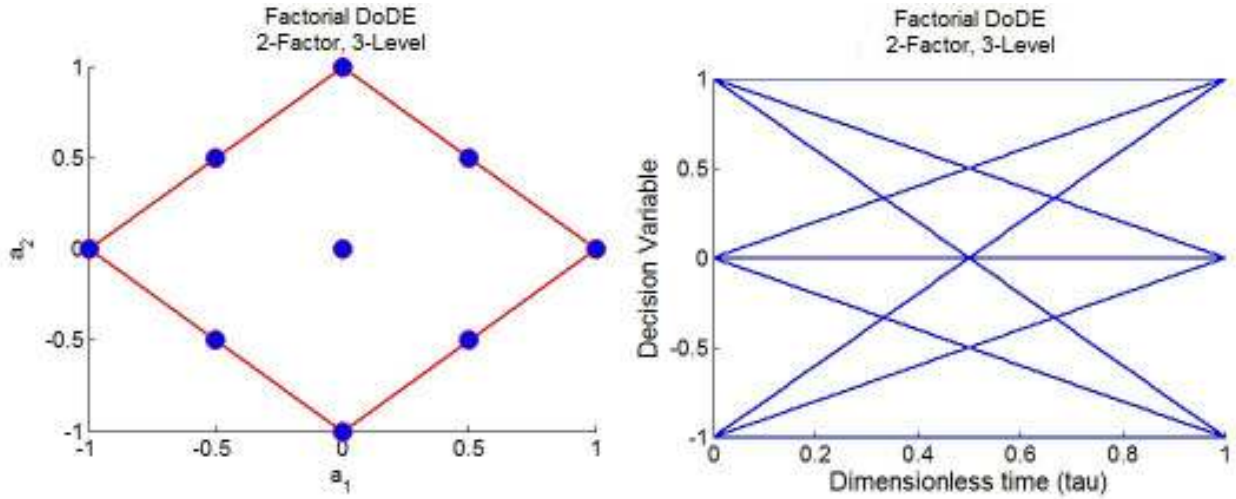

Figure 6. Factorial DoDE for 2-Factor, 3-Level

3.3 Design Space (3-Factors)

Quadratic temperature profiles are examined once a third dynamic factor is introduced. The decision variable function and its first derivative then becomes:

$$\begin{aligned} u(\tau) &= (a_1 + a_2 + a_3) + (-2a_2 - 6a_3)\tau + 6a_3\tau^2; \quad \tau \in [0,1] \\ u'(\tau) &= (-2a_2 - 6a_3) + 12a_3\tau \end{aligned} \tag{19}$$

The root of $u'(\tau_e) = 0$ defines an extremum at τ_e and must also be considered when determining the constraints on the a_i's to maintain the temperature profile within the allowed range. The following calculations lead to a set of 6 constraints.

$$u'(\tau_e) = 0 = (-2a_2 - 6a_3) + 12a_3\tau_e \Rightarrow \tau_e = \frac{a_2 + 3a_3}{6a_3}$$

$$-1 \le u(\tau_e) = a_1 - \frac{a_2^2 + 3a_3^2}{6a_3} \le 1 \text{ which can be written more usefully as:}$$

$$-a_3^2 \le a_1 a_3^2 - \frac{a_2^2 a_3}{6} - \frac{a_3^3}{2} \le a_3^2 \tag{20}$$

$$-1 \le u(0) = a_1 + a_2 + a_3 \le 1$$

$$-1 \le u(1) = a_1 - a_2 + a_3 \le 1$$

The addition of the two non-linear constraints above adds a bit of complexity to the visualization of the allowable region for the design space, so the following figures show a few different views of the 6 constraints. Intersections between constraints have been parameterized so that the edges are shown, however the details of the calculation of these edges is not presented.

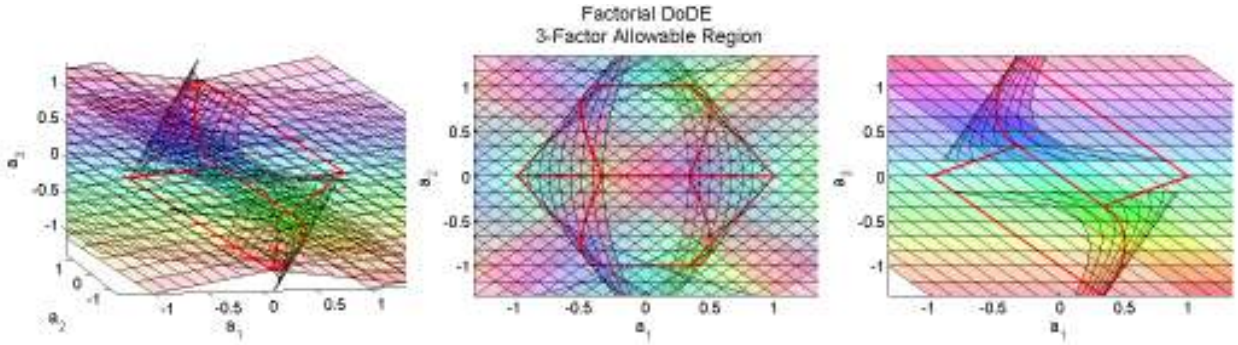

Figure 7. Allowable Region for the Design Space (3-Factor DoDE)

From the allowable region, experiments can be designed to fill the maximum amount of the design space. Because this allowable region is more complex than the 2-factor case and the values of the inputs may not readily be available, the factor inputs are shown in the following table for the 2-level and 3-level designs. The following figures show the resulting design points and temperature profiles (dimensionless) for these designs. Only the edges of the design space, and not the constraint planes, are shown.

Table 2. Factorial DoDE for 3-Factor, 2-Level

	Inputs		
Experiment	a_1	a_2	a_3
1	1	0	0
2	-1	0	0
3	0.33	0	-1.33
4	-0.33	0	1.33
5	0.33	1	-0.33
6	0.33	-1	-0.33
7	-0.33	1	0.33
8	-0.33	-1	0.33

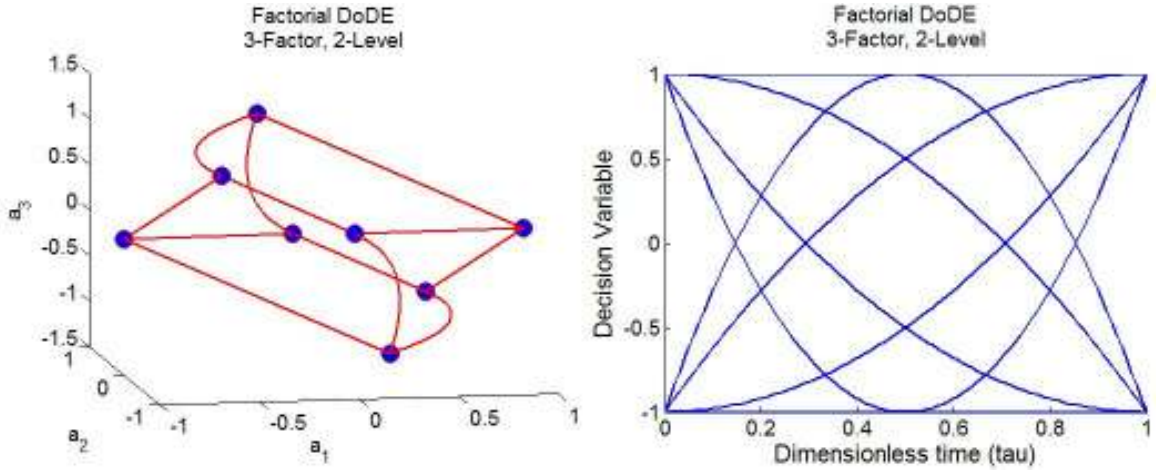

Figure 8. Factorial DoDE for 3-Factor, 2-Level

Table 3. Factorial DoDE for 3-Factor, 3-Level

	Inputs		
Experiment	a_1	a_2	a_3
1	0	0	0
2	1	0	0
3	-1	0	0
4	0.33	0	-1.33
5	-0.33	0	1.33
6	0.33	1	-0.33
7	0.33	-1	-0.33
8	-0.33	1	0.33
9	-0.33	-1	0.33
10	0.33	0	0.67
11	0.67	0	-0.67
12	-0.33	0	-0.67
13	-0.67	0	0.67
14	0.67	-0.5	-0.17
15	0	-1	0
16	-0.47	-0.5	0.97
17	0.17	-0.5	0.33
18	-0.67	-0.5	0.17
19	-0.67	0.5	0.17
20	-0.47	0.5	0.97
21	0.17	0.5	0.33
22	0	1	0
23	0.67	0.5	-0.17
24	-0.17	-0.5	-0.33
25	-0.17	0.5	-0.33
26	0.47	-0.5	-0.97
27	0.47	0.5	-0.97

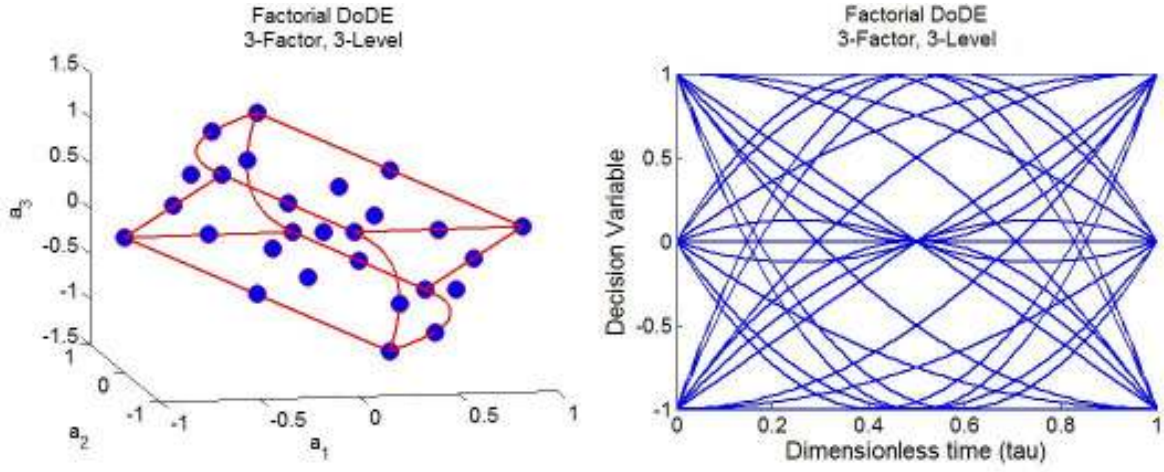

Figure 9. Factorial DoDE for 3-Factor, 3-Level

3.4 Results (T_{ref} = 35°C)

Only the results for the 3-Factor, 3-Level design will be discussed here. My results for the other factor-level combinations for this problem have been published by Georgakis (2008).[7] Those results were generated earlier using a different set of design points for the 3-factor cases, so the conclusions will not be exactly the same. Importantly, in that paper it was shown that standard DoE yielded an optimum of 62.2% conversion where as 3-factor, 3-level DoDE yielded 77.5% conversion. Two important improvements besides the newer design points (Table 3) have been implemented for this paper: statistical model term removal (similar to stepwise regression) and induced error.

Normally distributed induced error of 0, 1, 2, 5 and 10% was added to the experimental response, and was then used to make response surfaces and perform optimization. Due to the random nature of this induced error, the entire DoDE process was repeated 100 times for each level of induced error. This allowed for the standard deviation to be calculated for the DoDE methodology at each error level. Additionally, the average width of the confidence interval on the model prediction at x_{opt} is included to show the prediction accuracy. Simulated runs were performed at the optimum inputs, x_{opt}, to determine the "real-world" result using DoDE to optimize this process.

As described by Georgakis (2008), the exactly solved ideal temperature profile for this process was obtained using sequential quadratic programming.[7] This approach divides the time span into finite elements, for each of which the method of orthogonal collocations is used to convert the system of ODEs into a set of algebraic equations.[9] The exactly solved temperature profile is shown in the following figure along with concentration curves. For a batch time of 2.5 hours the true model based optimum conversion of reactant A was calculated this way to be 77.68%.

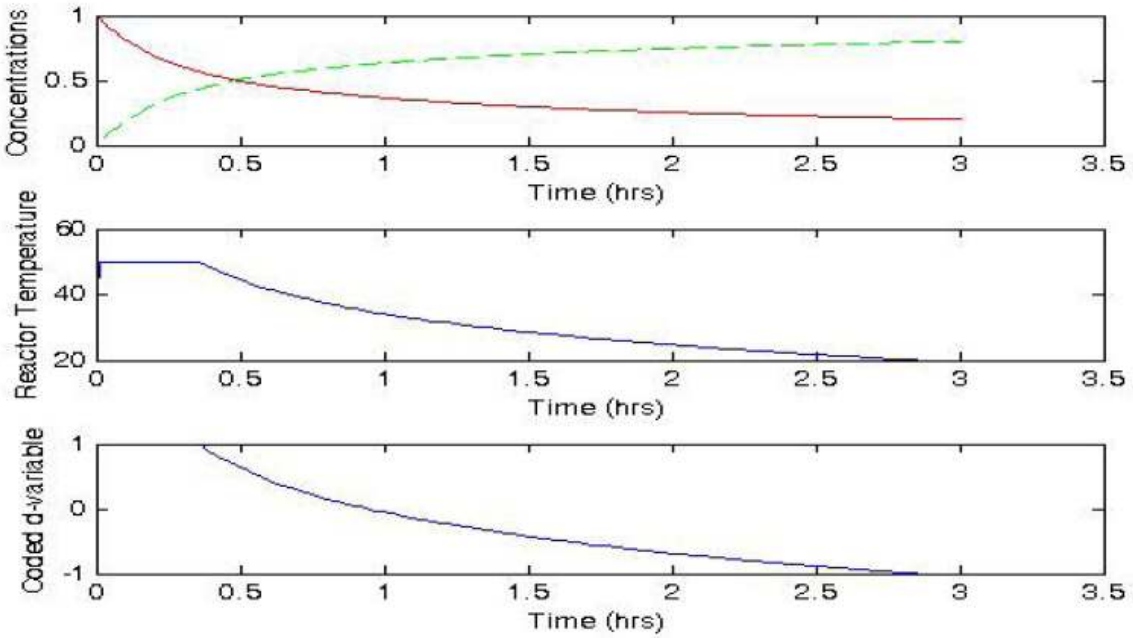

Figure 10. Exactly Solved (Sequential Quadratic Programming) Temperature Profile[7]

At each error level the DoDE determined optimal inputs (x_{opt}) were determined with 100 replicates. These optimal inputs are shown in Figure 11. As the amount of induced error increases, so does the spread in the DoDE predicted optimum. Term removal seems to reduce this spread slightly. The following tables summarize the DoDE predicted conversions versus the "real-world" results at x_{opt}. Standard deviations and mean confidence interval widths are included.

Table 4. DoDE Optimum Conversions with Induced Error (T_{ref} = 35°C) with No Term Removal

	Mean DoDE Prediction		Mean Result at x_{opt}	
Error	Conversion	StDev	Conversion	StDev
0%	77.39% ± 0.629%	0.00%	77.01%	0.00%
1%	77.36% ± 0.980%	0.16%	77.00%	0.06%
2%	77.41% ± 1.755%	0.34%	76.96%	0.13%
5%	77.61% ± 3.838%	0.76%	76.61%	0.50%
10%	78.60% ± 8.072%	1.60%	75.82%	1.22%

Table 5. DoDE Optimum Conversions with Induced Error (T_{ref} = 35°C) with Term Removal

	Mean DoDE Prediction		Mean Result at x_{opt}	
Error	Conversion	StDev	Conversion	StDev
0%	77.37% ± 0.580%	0.00%	77.00%	0.00%
1%	77.19% ± 0.928%	0.21%	76.98%	0.04%
2%	77.24% ± 1.556%	0.38%	76.86%	0.23%
5%	77.57% ± 3.546%	0.72%	76.68%	0.44%
10%	78.15% ± 6.813%	1.49%	76.53%	0.97%

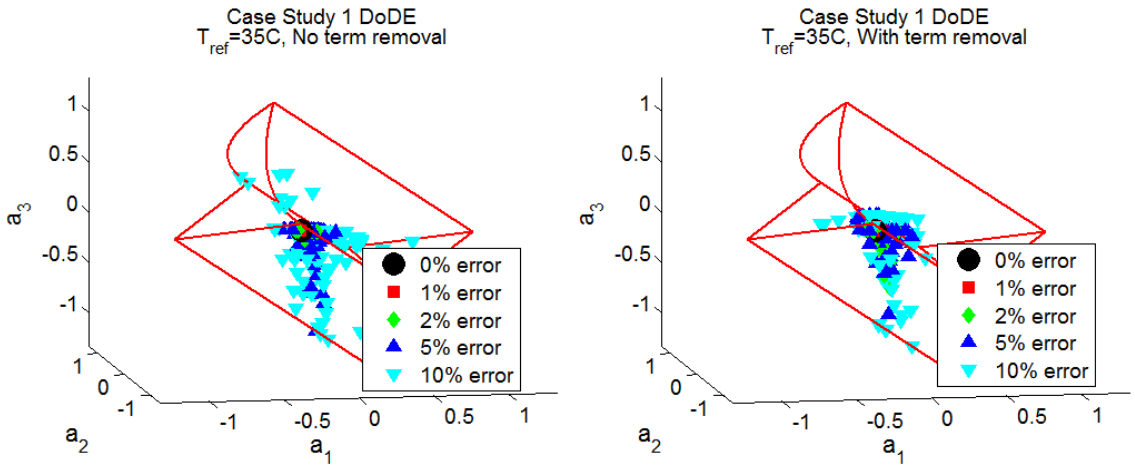

Figure 11. DoDE Determined Optimum Inputs for Different Induced Error (T_{ref} = 35°C)

Reducing the model by performing term removal increased the statistical accuracy of the DoDE model predictions by reducing the confidence interval width near x_{opt}. Induced experimental error seemed to not effect the DoDE model predicted optimum conversion, but it did have a small effect on the actual conversion if an experiment were performed with inputs x_{opt}. The following figure shows that term removal had little effect on the shape of the DoDE predicted optimum temperature profile. The two temperature profiles are nearly overlapping so only the term removal curve is shown. For comparison, the true kinetic model based optimum temperature profile is included as reference.

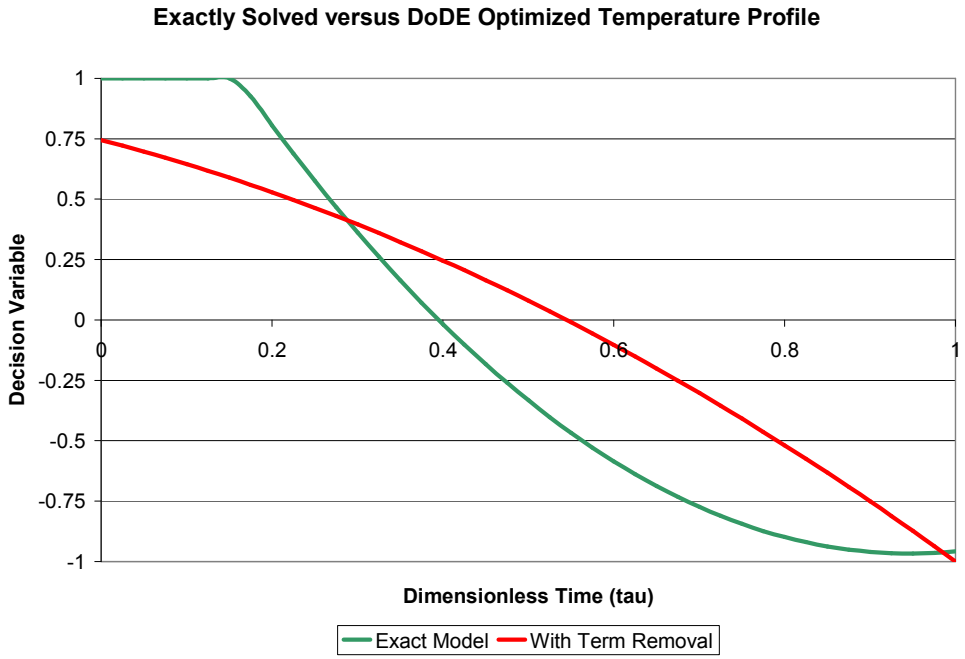

Figure 12. DoDE Predicted Optimum Temperature Profile versus Exactly Solved Profile

3.5 Results (T_{ref} = 30°C)

The flat region, at maximum temperature, for the exactly solved temperature profile is presumed to increase in length for the T_{ref} = 30ºC case. Maintaining this maximum temperature longer in the start of the process would give a higher conversion. This DoDE design does not incorporate a flat region so it was predicted that DoDE would have reduced optimization power at these temperatures. It should be noted that it is possible to incorporate a "switch-point" into a DoDE design such that flat regions and other unique regions in the ideal temperature profile can be optimized for. This has been shown by Georgakis (2008) for a fermentation process in which there are unique substrate feed profiles for the growth and production phase.[7]

Performance is only slightly reduced for this temperature range as shown in the following figures and tables. Spread of the graphed x_{opt}'s is reduced along with the confidence interval width of the DoDE model predictions when terms are removed. Again, the removal of terms from the model had little effect on the shape of the DoDE optimized temperature profile. The two profiles are nearly identical. Overall, the conversions obtained for this temperature range are not much lower than the T_{ref} = 35ºC case. To further investigate, a still lower temperature range was examined and is explained in the next section.

Table 6. DoDE Optimum Conversions with Induced Error (T_{ref} = 30°C) with No Term Removal

	Mean DoDE Prediction		Mean Result at x_{opt}	
Error	Conversion	StDev	Conversion	StDev
0%	77.04% ± 0.662%	0.00%	76.91%	0.00%
1%	77.04% ± 1.019 %	0.16%	76.88%	0.05%
2%	77.06% ± 1.692%	0.32%	76.79%	0.18%
5%	77.31% ± 4.116%	0.87%	76.47%	0.39%
10%	78.32% ± 8.276%	1.99%	75.79%	0.95%

Table 7. DoDE Optimum Conversions with Induced Error (T_{ref} = 30°C) with Term Removal

	Mean DoDE Prediction		Mean Result at x_{opt}	
Error	Conversion	StDev	Conversion	StDev
0%	77.10% ± 0.578%	0.00%	76.91%	0.00%
1%	77.09% ± 0.938%	0.13%	76.90%	0.02%
2%	77.18% ± 1.586%	0.35%	76.86%	0.12%
5%	77.35% ± 3.506%	0.74%	76.74%	0.34%
10%	77.23% ± 6.406%	1.48%	76.29%	1.35%

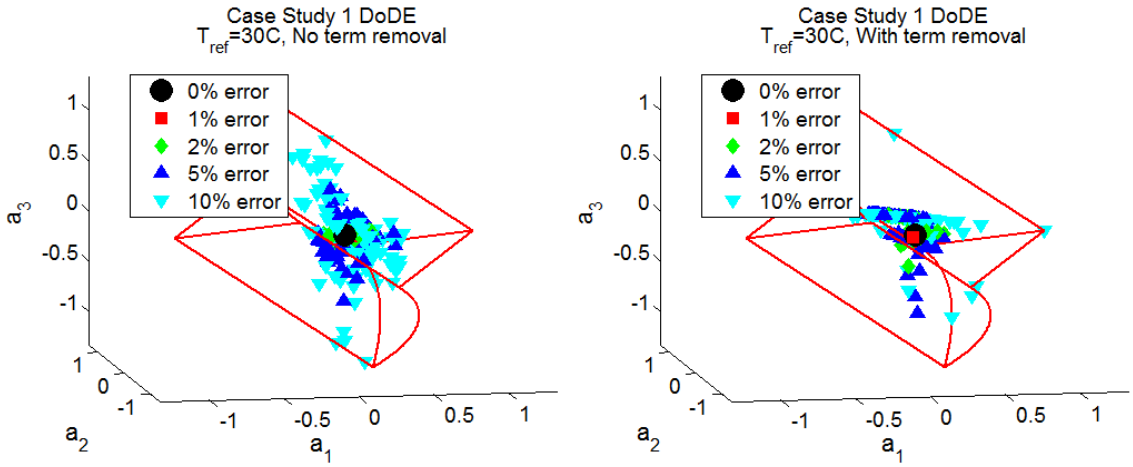

Figure 13. DoDE Determined Optimum Inputs for Different Induced Error (T_{ref} = 30°C)

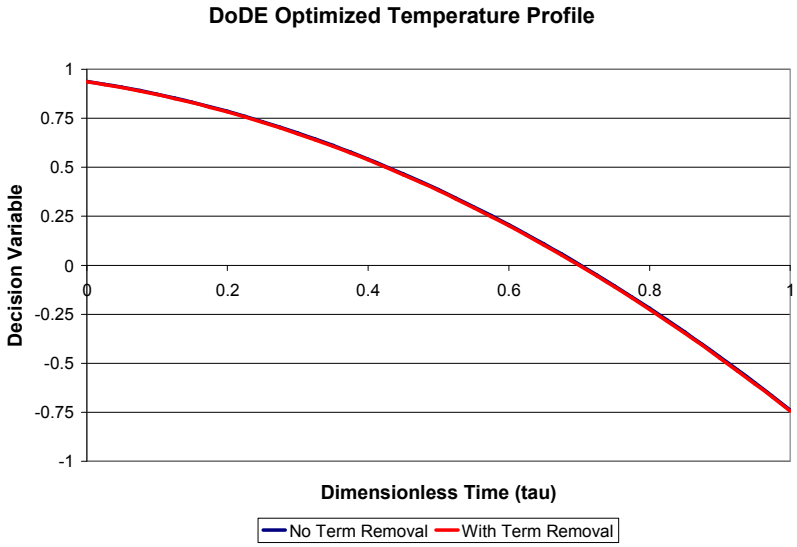

Figure 14. DoDE Predicted Optimum Temperature Profiles (T_{ref} = 30°C)

3.6 Results (T_{ref} = 20°C)

This temperature range produced different results as shown in the following tables and figures. The response surface model optimal temperature profiles are nearly linear with shallow slopes. Conversion in this temperature range is greatly reduced.

Table 8. DoDE Optimum Conversions with Induced Error (T_{ref} = 20°C) with No Term Removal

	Mean DoDE Prediction		Mean Result at x_{opt}	
Error	Conversion	StDev	Conversion	StDev
0%	74.12% ± 0.945%	0.00%	74.73%	0.00%
1%	74.17% ± 1.435%	0.25%	74.66%	0.26%
2%	74.23% ± 2.519%	0.57%	74.59%	0.43%
5%	74.62% ± 5.537%	1.42%	74.52%	0.62%
10%	75.62% ± 11.262%	2.28%	74.12%	1.43%

Table 9. DoDE Optimum Conversions with Induced Error (T_{ref} = 20°C) with Term Removal

	Mean DoDE Prediction		Mean Result at x_{opt}	
Error	Conversion	StDev	Conversion	StDev
0%	74.06% ± 0.840%	0.00%	74.52%	0.00%
1%	74.36% ± 1.454%	0.38%	74.28%	0.37%
2%	74.66% ± 2.269%	0.53%	74.26%	0.41%
5%	74.94% ± 4.686%	1.29%	74.44%	0.61%
10%	76.29% ± 8.779%	3.03%	74.11%	0.91%

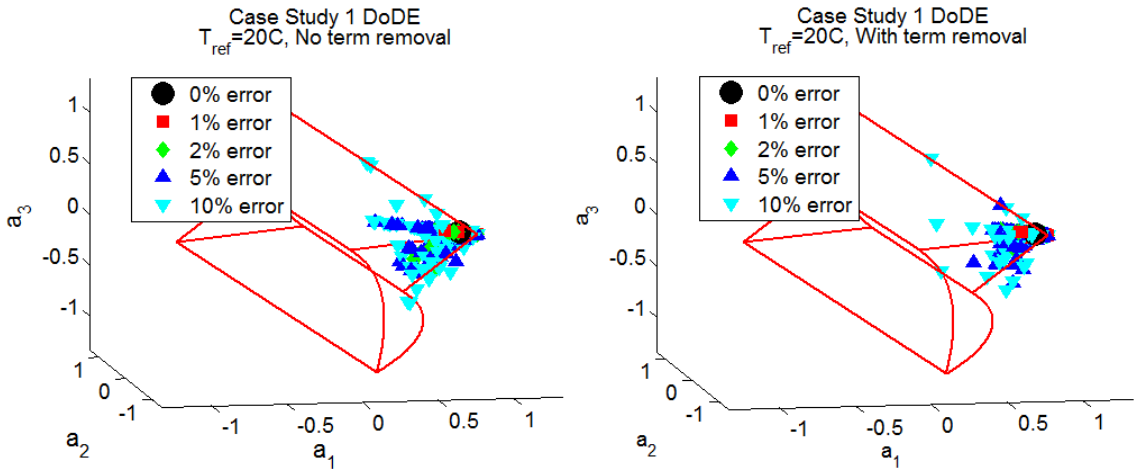

Figure 15. DoDE Determined Optimum Inputs for Different Induced Error (T_{ref} =20°C)

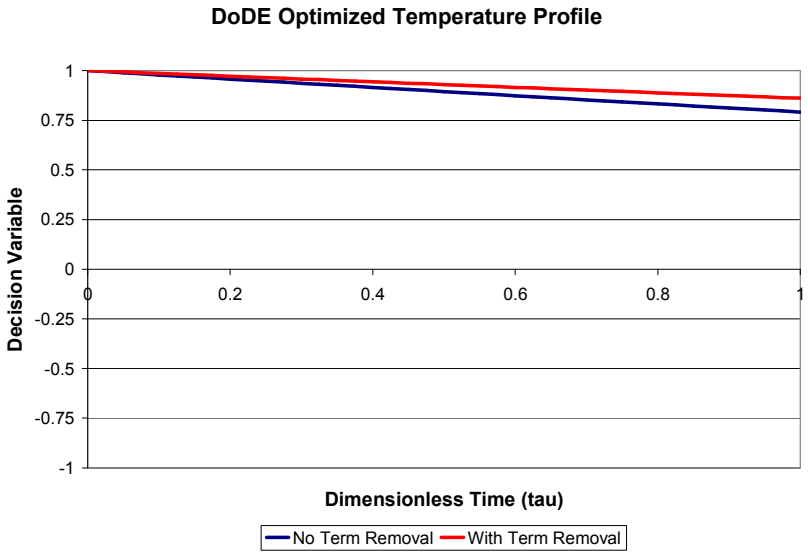

Figure 16. DoDE Predicted Optimum Temperature Profiles (T_{ref} = 20°C)

3.7 Conclusions

This case study revealed that model term removal was an important process and should be included into the DoDE methodology. The DoDE methodology was shown to optimize this process high conversion (77%) comparable to the maximum (true model exactly solved temperature profile) obtainable conversion of 77.68% at T_{ref} = 35°C. Coming this close to the true optimum was indeed shocking, but more complicated kinetics had to be examined.

4. Case Study 2: 2nd Order Reversible Reaction

4.1 Problem Definition

The simple process examined in the first case study was modified slightly to examine an alternative kinetic model that introduced nonlinear dynamics in the reaction model. Second-order reaction kinetics was imposed on the forward reaction of Equation 14 and the modified kinetic constants are shown below. All of the other kinetics remained unchanged.

$$r = k_1 C_A^2 - k_2 C_B$$
$$k_{1,0} = 1.76 \cdot 10^7 \ \mathrm{L \cdot gmol^{-1} \cdot hr^{-1}} \tag{21}$$

4.2 Constraints

For simplicity, only the T_{ref} = 35ºC case was examined and only the term removal DoDE results will be shown. The other constraints are the same as in the last case study.

4.3 Results

In the following figure the spread in the predicted optimum inputs graphically appeared to be much higher, but the standard deviation in the predicted and "real-world" result conversions remained low. In this case study, the best conversion from experiment was 64%. As shown in the following table, however, the DoDE methodology was not able to produce higher conversion. It can often happen that the maximum conversion is actually one of the tested experiments, so this is not viewed as a problem with the methodology.

Table 10. Case Study 2 DoDE Optimum Conversions with Induced Error and Term Removal

	Mean DoDE Prediction		Mean Result at x_{opt}	
Error	Conversion	StDev	Conversion	StDev
0%	64.08% ± 0.626%	0.00%	63.94%	0.00%
1%	64.14% ± 1.234%	0.30%	63.90%	0.20%
2%	64.29% ± 2.246%	0.51%	63.78%	0.40%
5%	65.39% ± 4.875%	1.17%	63.32%	1.14%
10%	66.65% ± 9.080%	2.85%	62.42%	1.63%

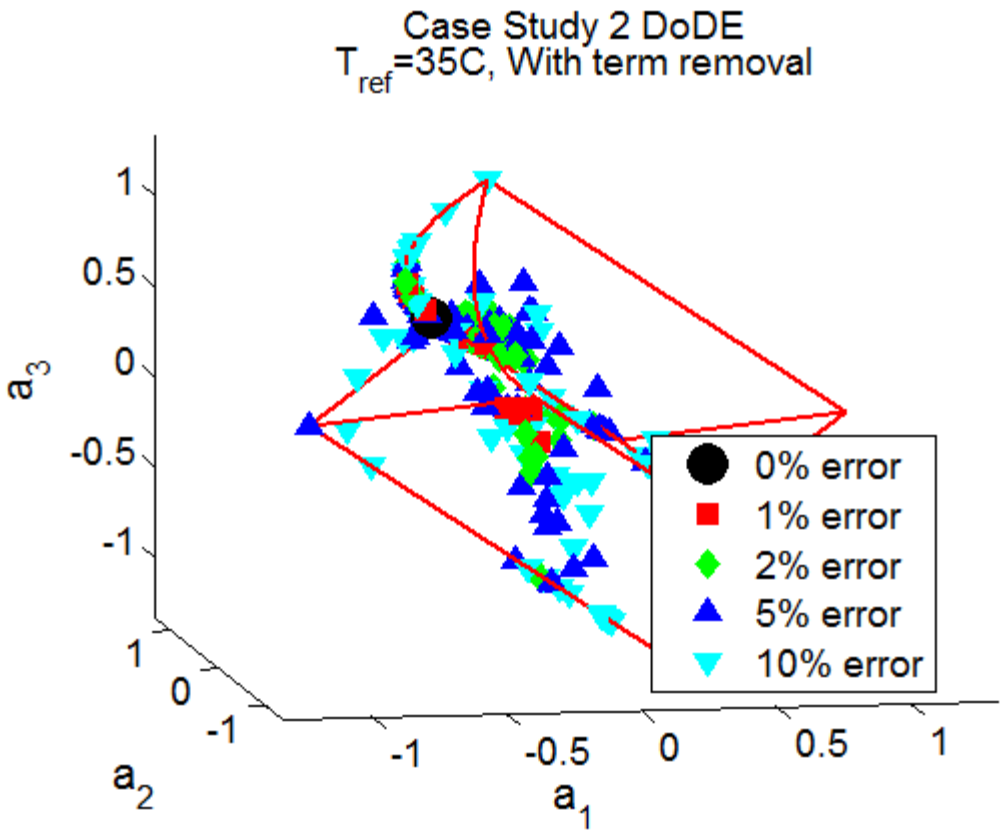

Figure 17. Case Study 2 DoDE Determined Optimum Inputs for Different Induced Error

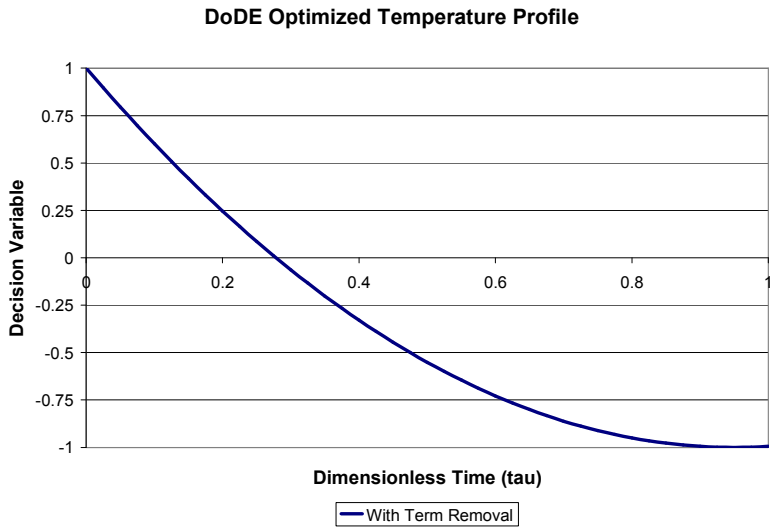

Figure 18. Case Study 2 DoDE Predicted Optimum Temperature Profile

4.4 Conclusions

Increasing the complexity of the process reaction kinetics only slightly reduced the DoDE accuracy. Again, the mean result when the DoDE predicted x_{opt} experiment was performed increased only slightly with induced experimental error. Even up to 10% error induced onto the measurements, the optimization still leads to an improved conversion. More complex kinetics as well as applying DoDE to control another process variable (instead of reactor temperature) were desired and examined in the next case study.

5. Case Study 3: Fed Batch Reaction with Inhibition and Side Reaction

5.1 Problem Definition

In this case study classical DoE was compared with DoDE in the optimization of a fed-batch process to produce compound C. The following kinetics was present. Note that reaction 1 has inhibition with reactant B.

$$A + B \longrightarrow C; \qquad r_1 = \frac{k_1 C_A C_B}{1 + k_2 C_B^2}$$
$$C \longrightarrow D; \qquad r_2 = k_2 C_C \tag{22}$$
$$k_1 = 50\ L \cdot gmol^{-1} hr^{-1}, \quad k_2 = 9\ hr^{-1}, \quad k_3 = 2\ hr^{-1}$$

Classical DoE factors for this fed batch reactor were defined as:

B_t = total amount (gmol) of B fed to the reactor
α = fraction of B initially added to the reactor at t=0
t_b = batch time (hr)

With both the stock solution of A and B having a concentration of 1M, the initial conditions of the reactor can be defined as:

$n_{A,0}$ = 1 gmol
$n_{B,0} = B_t \cdot \alpha$ [=] gmol (23)

$n_{C,0} = n_{D,0} = 0$ gmol
$V_0 = 1 + n_{B,0} / C_{B,0}$ [=] L, $C_{B,0} = 1$M

The decision variable, $u(\tau)$, is the flow rate of the remaining B solution into the reactor. For the classical DoE, this is a constant flow rate with time to add the remaining B solution in the batch time. This constant flow rate will be referred to as u_0 and is calculated as:

$$u_0 = \frac{B_t(1-\alpha)}{t_b \cdot C_{B,0}} \quad [=] \text{ L/hr} \tag{24}$$

When dynamic factors are included in the DoDE design, the flow rate of B can take on more complex shapes than just the linear profile for classical DoE. The dynamic form the design variable can then be written using Equation 9 and the shifted Legandre polynomials of Equation 10 as:

$$\begin{aligned} u(\tau) &= u_0(1 + w(\tau)) \\ w(\tau) &= a_1\varphi_1(\tau) + a_2\varphi_2(\tau) + a_3\varphi_3(\tau) + \dots \\ \tau &= t/t_b \end{aligned} \tag{25}$$

The response for these experiments will be the productivity of the reactor, calculated as the amount of C (gmol) produced per hour. This is shown below:

$$\text{Productivity} = \frac{n_{C,t=t_b}}{t_b} = \frac{n_{C,\tau=1}}{t_b} \tag{26}$$

5.2 Constraints

Minimal constraints are placed on the DoE design space. The examined region includes total B amounts from 1 to 2 gmol, fractions of B initially in the reactor from 0 to 100%, and batch times from 1 to 2 hours. These constraints on the static factors are shown in the following equation and remain unchanged in the DoDE design.

$$\begin{aligned} &1\text{ gmol} \le B_t \le 2\text{ gmol} \\ &0 \le \alpha \le 1 \\ &1\text{ hr} \le t_b \le 2\text{ hr} \end{aligned} \tag{27}$$

For the DoDE design, the first constraint examined was a flow constraint. If $w(\tau)$ is constrained above -1 for all τ between 0 and 1, then $u(\tau)$ will be ensured to be positive. This is important to maintain the flow rate non-negative and reasonable. Placing an upper constraint on $w(\tau)$ of 1 will maintain that the pump maximum capacity is no more than twice what is required in the DoE design. The constraints placed on $w(\tau)$ are then the same constraints placed on the decision variable from the previous case studies described in Equation 20.

Unique to this case study is the requirement on the volume of the B stock solution that is added. To be able to make comparisons between the DoE results and the DoDE results, the following equation must be held true:

$$\begin{aligned} &u_0 = \int_0^1 u(\tau)d\tau = \int_0^1 u_0\left(1 + w(\tau)\right)d\tau \\ &\Rightarrow 1 = \int_0^1 1 + w(\tau)d\tau \\ &\Rightarrow 0 = \int_0^1 w(\tau)d\tau \end{aligned} \tag{28}$$

Only looking at the first three dynamic factors, this becomes:

$$\begin{aligned} &0 = \int_0^1 a_1\varphi_1(\tau) + a_2\varphi_2(\tau) + a_3\varphi_3(\tau)d\tau \\ &0 = \int_0^1 \left(a_1 + a_2 + a_3\right) + \left(-2a_2 - 6a_3\right)\tau + 6a_3\tau^2 d\tau \\ &0 = \left(a_1 + a_2 + a_3\right) + \left(-a_2 - 3a_3\right) + 2a_3 \\ &0 = a_1 \end{aligned} \tag{29}$$

Thus the first dynamic factor is not required, as the constraints limit its value to zero. With $a_1 = 0$, the constraints from Equation 20 can be rewritten as:

$$\begin{aligned} &-1 \le u(0) = a_2 + a_3 \le 1 \\ &-1 \le u(1) = -a_2 + a_3 \le 1 \end{aligned} \tag{30}$$

Non-linear constraints are thus not required in this problem even for the DoDE design space. The design space for the two dynamic factors is similar to that shown in figure 6, except that the a_1 axis is replaced by a_3.

5.3 Results

DoE was performed using 0%, 1%, 2%, 5%, and 10% induced error onto the final amount (gmol) of C produced. Each error level design was performed 100 times to get an idea of the distribution of this method's results. Productivity (gmol C/hr) was calculated using Equation 26. Term removal was used in the regression to create a response surface to predict the productivity from the static factors: B_t, α, and t_b. The results are shown in the following table:

Table 11. Case Study 3 DoE Optimum Productivity with Induced Error and Term Removal

	Mean DoE Prediction		Mean Result at x_{opt}	
Error	gmole C/hr	StDev	gmole C/hr	StDev
0%	0.3515 ± 0.0178	0.00%	0.3453	0.00%
1%	0.3516 ± 0.0181	0.18%	0.3453	0.00%
2%	0.3512 ± 0.0191	0.42%	0.3453	0.00%
5%	0.3532 ± 0.0251	1.30%	0.3455	0.12%
10%	0.3500 ± 0.0407	2.35%	0.3456	0.49%

Due to the nature of the objective function, productivity, even with high induced error the DoE method converged on an optimum of low batch time, low initial concentration of B, and low amount of B added. This is shown in the following figure:

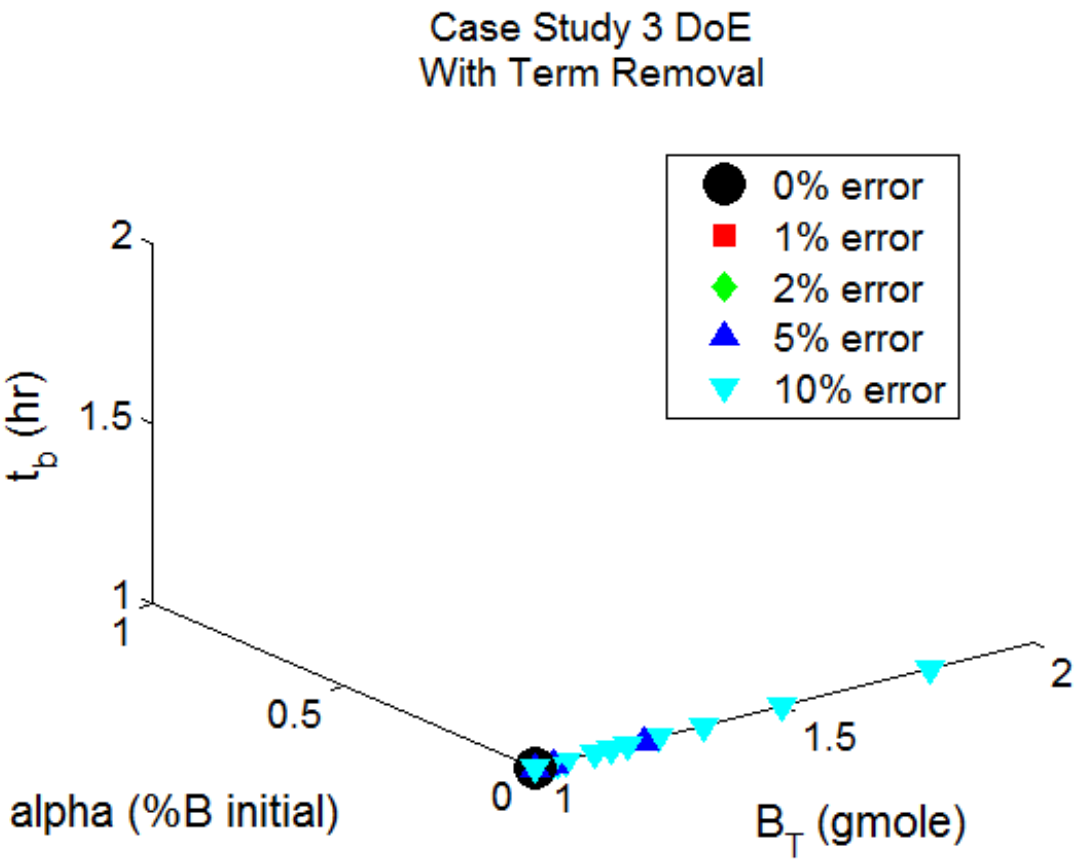

Figure 19. Case Study 3 DoE Determined Optimum Inputs for Different Induced Error

Using this optimum found from classical DoE as a starting point, a DoDE design was created to optimize the feed rate profile. For a 3-level, 2-factor design shown in figure 6, this required only 9 (= 3^2) additional experiments. These experiments would vary the values of a_2 and a_3 but maintain that: t_b = 1 hr, B_t = 1 gmol, and α = 0. The results of this are shown in the following table:

Table 12. Case Study 3 DoDE Optimum Productivity with Induced Error and Term Removal

	Mean DoE Prediction		Mean Result at x_{opt}	
Error	gmole C/hr	StDev	gmole C/hr	StDev
0%	0.4214 ± 0.0018	0.00%	0.4214	0.00%
1%	0.4207 ± 0.0107	0.44%	0.4214	0.00%
2%	0.4192 ± 0.0204	0.74%	0.4209	0.33%
5%	0.4190 ± 0.0462	1.67%	0.4190	0.71%
10%	0.4224 ± 0.0802	3.76%	0.4101	1.70%

Regardless of the amount of error induced, the optimum dynamic factors were quite consistent as shown in the following figure. Nearly all of the optimum have a_2 = -1 and a_3 = 0,

resulting in a linear $w(\tau)$ and thus a feed rate with a steeper slope than the DoE optimum. A comparison of the optimized feed profiles for the DoE and DoDE is shown in here as well. Curves for the instantaneous feed rate and the cumulative volume added to the reactor are shown.

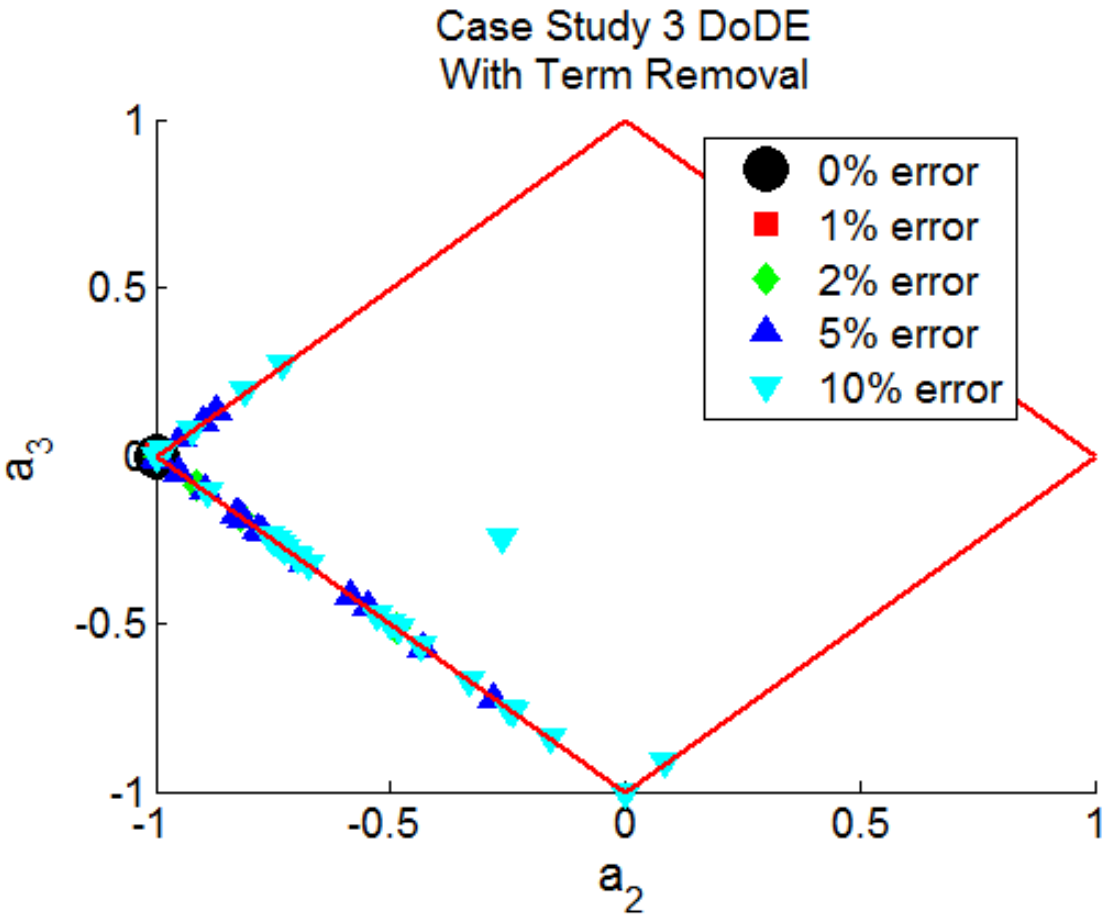

Figure 20. Case Study 3 DoDE Determined Optimum Inputs for Different Induced Error

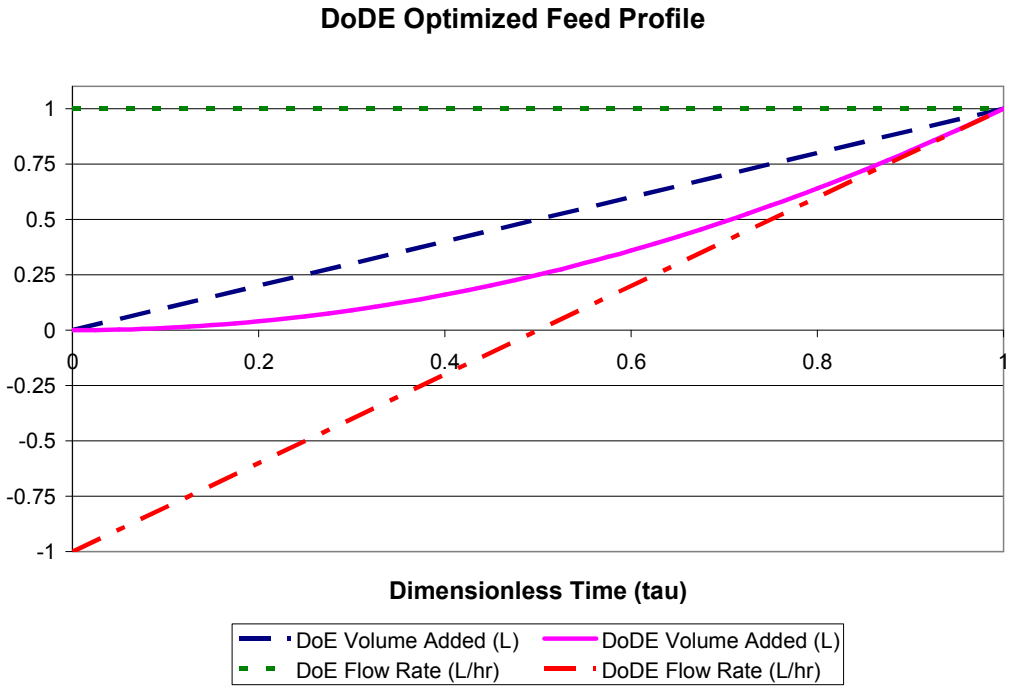

Figure 21. DoE and DoDE Optimized Feed Profiles

5.4 Conclusions

For the DoE design, 27 experiments were run and the optimized process produced a productivity of 0.3515 gmol C per hour. In the DoDE design, an additional 9 experiments were run and the optimized process produced a productivity of 0.4214 gmol C per hour. At the cost of performing 33% more experiments, the DoDE methodology produced a shocking 20% improvement in productivity. This case study demonstrated that DoDE can build off of DoE results to further optimize the process for dynamic variables.

6. Conclusions

Design of Dynamic Experiments (DoDE) can be applied to cheaply and quickly optimize operations which are affected by dynamic variables. The goal of this research was to develop the techniques to apply DoDE to rapidly optimize batch operations. Several kinds of reaction stoichiometry were simulated with varied degrees of kinetic complexity. These simulations provided the "data" for the DoDE optimization approach. The results were compared with the ideal (model-based) optimal operation which was calculated using sequential quadratic

programming (SQP). The DoDE methodology was able to produce high conversion and productivity processes.

7. Sources

[1] Fisher, R.A. (1935). The Design of Experiments. (8th ed., 1966) New York: Hafner Press.
[2] Lindman, H. R. (1974). Analysis of Variance in Complex Experimental Designs. San Francisco: W. H. Freeman & Co.
[3] Hardin, R. H., Sloane, N. J. A. "A New Approach to the Construction of Optimal Designs." Journal of Statistical Planning and Inference, Vol. 37 (1993), 339–369.
[4] Draper, N.R. and Smith, H. Applied Regression Analysis. Wiley Series in Probability and Statistics (1998).
[5] "Linear Regression." Wikipedia.org. http://en.wikipedia.org/wiki/Linear_regression (April 26, 2009).
[6] Moler, Cleve. MATLAB. Vers. 7.5. Computer software. The MathWorks. 2007.
[7] Georgakis, C. "A Model-Free Methodology for the Optimization of Batch Processes: Dynamic Design of Experiments." 2008. Patent Pending.
[8] Rippin, D. W. T.. "Simulation of a Single- and Multiproduct Batch Chemical Plants for Optimal Design and Operation." Computers & Chemical Engineering (1983), 7(3), 137-156.
[9] Biegler, L. T.. "Solution of Dynamic Optimization Problems by Successive Quadratic Programming and Orthogonal Collocation." Computers & Chemical Engineering (1984), 8(3-4), 243-247.

8. Appendix

8.1 Case Study 1 MATLAB Files

```
% DoDE optimization for the batch problem by multiple linear regression.
% Constained non-linear optimization
% Reaction2 reversible batch reaction with temperature as decision variable.
% by W. Alex Marvin April 5, 2009

% Reversible A<-->P, forward = 1st-order, reverse = 1st-order

% Induced Experimental Error (alphas) = 0%, 1%, 2%, 5%, 10%
% For the Order 2 Temperature Profile Case

% Define batch operating parameters
    reactionType='A==P';
    tb = 2.5; %Batch time in hours
    T_ref = 308; %Kelvin average temperature
    dT=15; %Kelvin allowed temperature fluctuation
    x0=[1, 0]; %initial mole fractions
    AE1toAE2 = 2; %the ratio of AE2 to AE1 (for AE1toAE2=2, AE2=2*AE1)
    df=2;
    modeltype=3;
    figure
% Define p-value cutoff for model terms
    pvalmin=0.05;

% Width of mymodel
    factors=3;

% Perform term removal? 1 = Yes
    termremoval = 1;

% Define internal variables
    % Shifted-Legandge Polynomials
        P0= @(t) 1;
        P1= @(t) -(2*t-1);
        P2= @(t) 6*t^2-6*t+1;
        P3= @(t) -(20*t^3-30*t^2+12*t-1);
        P4= @(t) 70*t^4-140*t^3+90*t^2-20*t+1;
        P5= @(t) -(252*t^5-630*t^4+560*t^3-210*t^2+30*t-1);
        P6= @(t) 924*t^6-2772*t^5+3150*t^4-1680*t^3+420*t^2-42*t+1;
    % Dimensionless time span
        m=101;
        t=linspace(0,1,m);

    % Create the scaled design space
    %  Full factorial design space used for the Order 2 case
          P=[   0,   0,   0; %1
                1,   0,   0; %2
               -1,   0,   0; %3
              1/3,   0,-4/3; %4
```

```
            -1/3,    0, 4/3; %5
             1/3,    1,-1/3; %6
             1/3,   -1,-1/3; %7
            -1/3,    1, 1/3; %8
            -1/3,   -1, 1/3; %9
             1/3,    0, 2/3; %10
             2/3,    0,-2/3; %11
            -1/3,    0,-2/3; %12
            -2/3,    0, 2/3; %13
             2/3,-1/2,-1/6; %14
               0,   -1,   0; %15
          -2^.5/3,-1/2,(2*2^.5+3)/6; %16
             1/6,-1/2, 1/3; %17
            -2/3,-1/2, 1/6; %18
            -2/3, 1/2, 1/6; %19
          -2^.5/3, 1/2,(2*2^.5+3)/6; %20
             1/6, 1/2, 1/3; %21
               0,   1,   0; %22
             2/3, 1/2,-1/6; %23
            -1/6,-1/2,-1/3; %24
            -1/6, 1/2,-1/3; %25
           2^.5/3,-1/2,-(2*2^.5+3)/6; %26
           2^.5/3, 1/2,-(2*2^.5+3)/6; %27
             ];
    % Number of experiments
        l=length(P);
    % Final mole fractions
        PI=zeros(l,1);

% Initialization of Results to be exported in Excel
xlsResults1={};
xlsResults2={};
xlsResults3={};

% File name to save the data into Excel
if termremoval
    xlsFileName = 'DDoXSimpleRxnError_T35_removal';
else
    xlsFileName = 'DDoXSimpleRxnError_T35_noremoval';
end
figTitle = {'DDoX of a Simple Reaction with induced Error';'Tref = 35C'};

% Prepare the several DDoX runs
wf=zeros(size(t));
Temp=zeros(size(t));
for i=1:l
    % Call ODE solver with present wc values
    [t,x]=ode45(@DDoXSimpleRxnODE,t,x0,[],P(i,:),tb,T_ref,dT,AE1toAE2);
    PI(i)=x(length(x));
end

% Induce Error
alphas=[0,0.01,0.02,0.05,0.1];
xopts=zeros(length(alphas)*100-99,factors);
w=0;
```

```
p=0;
for i=1:length(alphas)
    if alphas(i)==0
        j=1;
    else
        j=100;
    end

    % Perform Regression and Optimization on 100 replicates
    ConversionMax=zeros(j,1);
    ConversionExact=zeros(j,1);
    ConversionExperi=zeros(j,1);
    for k=1:j
        e=randn(l,1);
        PIe=PI.*(1+alphas(i)*e);
        p=p+1;

        % Regression model form of the PI Response Surface
        mymodel= [ 0 0 0;
                   1 0 0;
                   0 1 0;
                   0 0 1;
                   1 1 0;
                   1 0 1;
                   0 1 1;
                   1 1 1;
                   2 0 0;
                   0 2 0;
                   0 0 2;
                   2 1 0;
                   2 0 1;
                   1 2 0;
                   0 2 1;
                   1 0 2;
                   0 1 2];

        % Perform Regression
        stats=regstats(PIe,P,mymodel,{'beta','fstat','tstat','covb'});
        b=stats.beta; %Model Coefficients
        f = stats.fstat;
        tstats=stats.tstat;
        covb=stats.covb;

        % Remove high p-value terms
        [Y,I]=max(tstats.pval);
        removed=[];
        if termremoval
            while Y>pvalmin
                removed=[removed,I];
                CoefTemp=[];
                [terms,blah]=size(mymodel);
                mymodel=[mymodel(1:I-1,:);mymodel(I+1:terms,:)];

stats=regstats(PIe,P,mymodel,{'beta','rsquare','r','standres','fstat','tstat'
});
```

```
            b=stats.beta; %Model Coefficients
            f = stats.fstat;
            tstats=stats.tstat;
            [Y,I]=max(tstats.pval);
        end
    end

    % Constrained Optimization of the Response Surface Function
    [xopt,PIMax]=DDoXOptimize(b,mymodel,df,modeltype);
    ConversionMax(k)=(x0(1)-PIMax)/x0(1);
    xopts(p,:)=xopt;

    % Add DDoX Predicted results to be exported to Excel file
    xlsResults2(k+w,2:3)={k,ConversionMax(k)};
    for ii=1:length(xopt)
        xlsResults2(k+w,6+ii)={xopt(ii)};
    end

    % Calculate and add DDoX conversion prediction confidence interval
    X = x2fx(P,mymodel);
    Z = x2fx(xopt,mymodel);
    sigma=(f.sse/(l-length(mymodel(:,1))))^0.5;
    t_val=tinv(1-0.05/2,l-length(mymodel(:,1))-1);
    width=t_val*sigma*(1+Z*(X'*X)^-1*Z')^0.5;
    xlsResults2(k+w,4)={sigma^2};
    xlsResults2(k+w,5)={ConversionMax(k)-width};
    xlsResults2(k+w,6)={ConversionMax(k)+width};

    % Using xopt calculate the Exact Model conversion
    [t,x]=ode45(@DDoXSimpleRxnODE,t,x0,[],xopt,tb,T_ref,dT,AE1toAE2);
    ConversionExact(k)=(x0(1)-x(length(x)))/x0(1);
    xlsResults2(k+w,10)={ConversionExact(k)};

    % Check to see if this conversion is within the DDoX confidence
    % interval
    ii = boolean(ConversionExact(k)>=(ConversionMax(k)-width) && ...
                 ConversionExact(k)<=(ConversionMax(k)+width));
    if ii
        xlsResults2(k+w,11)={'Yes'};
    else
        xlsResults2(k+w,11)={'No'};
    end

    % Add top conversion from experiment to be exported to Excel file
    ConversionExperi = (x0(1)-PIe)./x0(1);
    A=[ConversionExperi P];
    [Sorted,I] = sort( A(:,1),'descend');
    Sorted=A(I,:);
    for ii=1:length(Sorted(1,:))
        xlsResults2(k+w,11+ii)={Sorted(1,ii)};
    end

    % Add empty cells to account for removed model terms
    for ww=1:length(removed)
        www=removed(length(removed)-ww+1);
```

```
            tstats.pval=[tstats.pval(1:www-1);-
1;tstats.pval(www:length(tstats.pval))];
            tstats.beta=[tstats.beta(1:www-1);-
1;tstats.beta(www:length(tstats.beta))];
        end

        % Add DDoX Predicted model coefficients and statistics
        xlsResults2(k+w,16)={f.pval};
        for ii=1:length(tstats.pval)
            xlsResults2(k+w,16+ii)={tstats.pval(ii)};
            xlsResults2(k+w,34+ii)={tstats.beta(ii)};
        end
    end

    % Add results to be exported to Excel file
    xlsResults1(i,1:5)={alphas(i),mean(ConversionMax),std(ConversionMax),...
                        mean(ConversionExact),std(ConversionExact)};
    xlsResults2(w+1,1)={alphas(i)};
    w=w+j+1;
end

% Add the design space used information to be exported
for i=1:l
   for j=1:df+1
      xlsResults3(i,j)={P(i,j)};
   end
end

% Write data to Excel file
xlswrite(xlsFileName, xlsResults1, 'Sheet1', 'a7');
xlswrite(xlsFileName, xlsResults2, 'Sheet1', 'g7');
xlswrite(xlsFileName, xlsResults3, 'Sheet1', 'a17');

% Graph the resulting data points
hold on
plot3(xopts(1,1),xopts(1,2),xopts(1,3),'o',...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor','k',...
    'MarkerSize',15)
if length(alphas)>1
    plot3(xopts(2:101,1),xopts(2:101,2),xopts(2:101,3),'s',...
        'MarkerEdgeColor','r',...
        'MarkerFaceColor','r',...
        'MarkerSize',10)
    plot3(xopts(102:201,1),xopts(102:201,2),xopts(102:201,3),'d',...
        'MarkerEdgeColor','g',...
        'MarkerFaceColor','g',...
        'MarkerSize',10)
    plot3(xopts(202:301,1),xopts(202:301,2),xopts(202:301,3),'^',...
        'MarkerEdgeColor','b',...
        'MarkerFaceColor','b',...
        'MarkerSize',10)
    plot3(xopts(302:401,1),xopts(302:401,2),xopts(302:401,3),'v',...
        'MarkerEdgeColor','c',...
        'MarkerFaceColor','c',...
```

```
        'MarkerSize',10)
end
axis(4/3*[-1 1 -1 1 -1 1])
xlabel('a_1')
ylabel('a_2')
zlabel('a_3')

% % Add surfaces for inequalities
% [X,Y] = meshgrid(-4/3:1/6:4/3);
% nn=length(X);
% mm=ceil(nn/2);
% %X=b (P1)
% %Y=c (P2)
% %Z=a (P0)
% Z1=1-X-Y;
% Z2=-1-X-Y;
% Z3=1+X-Y;
% Z4=-1+X-Y;
% Z5=1+(X(1:(nn-mm),:).^2+3*Y(1:(nn-mm),:).^2)./(6*Y(1:(nn-mm),:)+eps);
% Z6=-1+(X((nn-mm+2):nn,:).^2+3*Y((nn-mm+2):nn,:).^2)./(6*Y((nn-
mm+2):nn,:)+eps);
%
% hold on
% surf(Z1,X,Y)
% colormap hsv
% alpha(.4)
% surf(Z2,X,Y)
% colormap hsv
% alpha(.4)
% surf(Z3,X,Y)
% colormap hsv
% alpha(.4)
% surf(Z4,X,Y)
% colormap hsv
% alpha(.4)
% surf(Z5,X(1:(nn-mm),:),Y(1:(nn-mm),:))
% colormap hsv
% alpha(.4)
% surf(Z6,X((nn-mm+2):nn,:),Y((nn-mm+2):nn,:))
% colormap hsv
% alpha(.4)

% Add Edges for intersections of 2 inequalities
xx=linspace(0,1)';
b=xx-1;
edg1=[1+b-(2*(b+1).^.5+b+2)/3,b,(2*(b+1).^.5+b+2)/3];
plot3(edg1(:,1),edg1(:,2),edg1(:,3),'r-','LineWidth',2)
edg2=[edg1(:,1),-edg1(:,2),edg1(:,3)];
plot3(edg2(:,1),edg2(:,2),edg2(:,3),'r-','LineWidth',2)
edg3=[-edg1(:,1),edg1(:,2),-edg1(:,3)];
plot3(edg3(:,1),edg3(:,2),edg3(:,3),'r-','LineWidth',2)
edg4=[-edg1(:,1),-edg1(:,2),-edg1(:,3)];
plot3(edg4(:,1),edg4(:,2),edg4(:,3),'r-','LineWidth',2)
edg5=[-1/3, -1, 1/3;
       1/3, -1,-1/3;
```

```
          1,  0,   0;
        1/3,  1,-1/3;
       -1/3,  1, 1/3;
         -1,  0,   0;
        1/3,  0,-4/3];
plot3(edg5(:,1),edg5(:,2),edg5(:,3),'r-','LineWidth',2)
edg6=[-1/3,  0, 4/3;
          1,  0,   0];
plot3(edg6(:,1),edg6(:,2),edg6(:,3),'r-','LineWidth',2)
edg6=[  -1,  0,   0;
       -1/3, -1, 1/3];
plot3(edg6(:,1),edg6(:,2),edg6(:,3),'r-','LineWidth',2)

% Change the camera view
view(-10,12)

% Add legend with the %error data markers
legend('0% error','1% error','2% error','5% error','10%
error','Location','Best')
title(figTitle)
hold off

% Format the font sizes of the figure
set(gca,'FontSize',14);
set(get(gca,'XLabel'),'FontSize',16);
set(get(gca,'YLabel'),'FontSize',16);
set(get(gca,'ZLabel'),'FontSize',16);
set(get(gca,'Title'),'FontSize',18);

% Save figure to file
saveas(gcf,[xlsFileName,'.fig'])


function dxdt=DDoXSimpleRxnODE(t,x,wc,tb,T_ref,dT,Ae1toAe2)
dxdt=zeros(2,1); %a column vector

% dimentional model paramaters
AE1= 10.0E03;   % in cal/gmol
AE2= Ae1toAe2*AE1;  % in cal/gmol
k10= 1.32E07;   % in i/hr
k20_old= 5.24E13;   % in i/hr
R= 1.98;        % Universal gas constant

% Change k20 so that k2_old(T_ref) = k2_new(T_ref)
AE2_old=2*AE1;
k20=k20_old*exp(-AE2_old/(R*T_ref)+AE2/(R*T_ref));

% dimentionLESS model parameters
alpha=dT/T_ref;         % dimensionless range of reactor temperature,
gamma1=AE1/(R*T_ref);   % dimension-LESS activation Energy of rxn 1
gamma2=AE2/(R*T_ref);   % dimension-LESS activation Energy of rxn 2

% define the Legandge polynomials
P0= @(t) 1;
P1= @(t) -(2*t-1);
```

```
P2= @(t) 6*t^2-6*t+1;
P3= @(t) 20*t^3-30*t^2+12*t-1;
P4= @(t) 70*t^4-140*t^3+90*t^2-20*t+1;
P5= @(t) 252*t^5-630*t^4+560*t^3-210*t^2+30*t-1;

% Coded decision variable w(t)
wc=[wc zeros(1,6-length(wc))];
w= @(t)
wc(1)*P0(t)+wc(2)*P1(t)+wc(3)*P2(t)+wc(4)*P3(t)+wc(5)*P4(t)+wc(6)*P5(t);
% dimensionless temperature
u= @ (t) 1+alpha*w(t);
%dimentional kinetic constants;
k1=k10*exp(-gamma1/u(t));
k2=k20*exp(-gamma2/u(t));
%dimentionless kinetic constants wrt tb
f1=k1*tb;
f2=k2*tb;
dxdt(1)=-f1*x(1)+f2*x(2);
dxdt(2)=+f1*x(1)-f2*x(2);


function [xopt,PIMax]=DDoXOptimize(b,mymodel,df)
% DDOX optimization for the batch problem by multiple linear regression.
% Constained non-linear optimization
% Simple reversible batch reaction with temperature as decision variable.
% by W. Alex Marvin April 9, 2009

% The 8 corner points plus the center point of the design space
P=[     0,   0,   0; %1
        1,   0,   0; %2
       -1,   0,   0; %3
      1/3,   0,-4/3; %4
     -1/3,   0, 4/3; %5
      1/3,   1,-1/3; %6
      1/3,  -1,-1/3; %7
     -1/3,   1, 1/3; %8
     -1/3,  -1, 1/3; %9
     ];
if df==0
    A=[1;-1];
    NONLCON=[];
else if df==1
        A=[1 1;-1 -1;1 -1;-1 1;1 0;-1 0];
        NONLCON=[];
    else if df==2
            A=[1 1 1;-1 -1 -1;1 -1 1;-1 1 -1;1 0 0;-1 0 0];
            NONLCON=@DDoxNonlcon2;
        else if df==3
                A=[1 1 1 1;-1 -1 -1 -1;1 -1 1 -1;-1 1 -1 1;1 0 0 0;-1 0 0 0];
                NONLCON=@DDoxNonlcon3;
            end
        end
    end
end
OPTIONS=optimset('GradConstr','on','GradObj','on');
```

```
FUN=@(x) DDoXFun(x,b,mymodel);
B=ones(1,length(A(:,1)));
Aeq=[];
Beq=[];
LB=[];
UB=[];
xopts=zeros(size(P));
PIMaxs=zeros(length(P),1);
for ii=1:9
    X0=P(ii,:); %initial inputs guess
    [xopt,PIMax]=fmincon(FUN,X0,A,B,Aeq,Beq,LB,UB,NONLCON,OPTIONS);
    xopts(ii,:)=xopt;
    PIMaxs(ii,1)=PIMax;
end

% Output the minimum xopt value with its related PIMax
[PIMax,I]=min(PIMaxs);
xopt=xopts(I,:);


function [PI,g]=DDoXFun(x,b,mymodel)
% Output the model predicted response
PI=x2fx(x,mymodel)*b;
[n,f]=size(mymodel);
% If required compute the gradient
if nargout>1
    g=zeros(1,f);
    for k=1:f
        modeltemp=mymodel;
        btemp=b;
        for l=1:n
           if modeltemp(l,k)>0
               if modeltemp(l,k)==2
                   btemp(l)=2*btemp(l);
               end
               modeltemp(l,k)=modeltemp(l,k)-1;
           else
               btemp(l)=0;
           end
        end
        g(k)=x2fx(x,modeltemp)*btemp;
    end
end
end


function [C,Ceq,gradc,gradceq]=DDoxNonlcon2(x)
% Last edited on 4/5/2009
% Computer the nonlinear constraints and gradients along them
a=x(1);
b=x(2);
c=x(3);
C(1)=a*c^2-b^2*c/6-c^3/2-c^2;
C(2)=-(a*c^2-b^2*c/6-c^3/2)-c^2;
Ceq=[];
```

```
if nargout>2
    gradc=[ c^2, -c^2;
            -b*c/3, b*c/3;
            2*a*c-b^2/6-3*c^2/2-2*c, -2*a*c+b^2/6+3*c^2/2-2*c];
    gradceq=[];
end
```

8.2 Case Study 2 MATLAB Files

The following m-files from case study 1 remained largely unchanged: `DDoXOptimize`, `DDoXFun`, and `DDoxNonlcon2`.

```
function dxdt=DDoXReaction2ODE(t,x,wc,tb,T_ref,dT,Ae1toAe2)
% For the Reaction A^2 = P

dxdt=zeros(2,1); % a column vector

% dimentional model paramaters
AE1= 10.0E03;   % in cal/gmol
AE2= Ae1toAe2*AE1;  % in cal/gmol
k10_old= 1.32E07;   % in 1/hr
k20_old= 5.24E13;   % in 1/hr
R= 1.98;        % Universal gas constant
xeq= 0.75;      % Concentration at which k10*xeq=k10_old
k10= k10_old/xeq;   % in 1/conc*hr

% Change k20 so that k2_old(T_ref) = k2_new(T_ref)
AE2_old=2*AE1; % This was when Ae1toAe2 = 2, the original reaction system
k20=k20_old*exp(-AE2_old/(R*T_ref)+AE2/(R*T_ref));

% dimentionLESS model parameters
alpha=dT/T_ref;         % dimensionless range of reactor temperature,
gamma1=AE1/(R*T_ref);   % dimension-LESS activation Energy of rxn 1
gamma2=AE2/(R*T_ref);   % dimension-LESS activation Energy of rxn 2

% define the Legandge polynomials
P0= @(t) 1;
P1= @(t) -(2*t-1);
P2= @(t) 6*t^2-6*t+1;
P3= @(t) 20*t^3-30*t^2+12*t-1;
P4= @(t) 70*t^4-140*t^3+90*t^2-20*t+1;
P5= @(t) 252*t^5-630*t^4+560*t^3-210*t^2+30*t-1;

% Coded decision variable w(t)
wc=[wc zeros(1,6-length(wc))];
w= @(t)
wc(1)*P0(t)+wc(2)*P1(t)+wc(3)*P2(t)+wc(4)*P3(t)+wc(5)*P4(t)+wc(6)*P5(t);
% dimensionless temperature
u= @ (t) 1+alpha*w(t);
%dimentional kinetic constants;
```

```
k1=k10*exp(-gamma1/u(t));
k2=k20*exp(-gamma2/u(t));
%dimentionless kinetic constants wrt tb
f1=k1*tb;
f2=k2*tb;
%dimentional reaction rates
r1=f1*x(1)^2;
r2=f2*x(2);
dxdt(1)=-r1+r2;
dxdt(2)=+r1-r2;
```

8.3 Case Study 3 MATLAB Files

```
% Alex Marvin
% Tufts University
% May 2, 2009

% DDoX Method (linear and quadratic dynamic feed variables)

% Fed Batch Reaction
% A + B --> C
% C --> D
% rate1 = k1*A*B/(1+k2*B^2)
% rate2 = k3*C

% Define batch operating parameters
    A0 = 1; %Initial moles of A (gmole)
    V0 = 1; %Initial reactor volume (L)
    B0 = 1; %Stock B concentration (gmole/L)

% Factors
%  Bt = total B to be added (range from 1 to 2 gmole)
%  alp = fraction of B initially in reactor (range from 0 to 1)
%  tb = batch time (range from 1 to 2 hr)
    Bt = 1; %gmole
    alp = 0;
    tb = 1; %hr
    Br = Bt*(1-alp);%remaining B to be added (gmole)

% Full factorial design space (3 level, 5 factor)
    % These values range from -1 to 1 and are the coded variable inputs
    % that are used for model fitting and optimization
    P = fullfact(3*ones(1,2))-2;
        % column 1 = coefficient infront of P1 (linear term)
        % column 2 = coefficient infront of P2 (quadratic term)

    % Scale column 4 and 5 (the dynamic variables)
    for i=1:length(P)
        if abs(P(i,1))+abs(P(i,2))==2
            P(i,1)=P(i,1)/2;
            P(i,2)=P(i,2)/2;
        end
```

```
    end


% Final mole fractions
    PI = zeros(length(P),4);

% Productivity Index (gmoleC/hour)
    J = zeros(length(P),1);

% Settings
    newrungraphs = 0; %Make graphs of experiments? 1 = Yes
    termremoval = 1; %Perform term removal? 1 = Yes
    xlsprintdata = 1; %Print data to excel? 1 = Yes
    pvalmin=0.05; %p-value cutoff for model terms

% Prepare the several DoX runs
for i=1:length(P)
    % Perform simulated experiments
    x0 = [A0, Bt*alp, 0, 0, V0+Bt*alp/B0]; %Initial concentrations (gmole/L)
and initial volume (L)
    [t,x]=ode45(@DDoXReaction3ODE3,[0 tb],x0,[],Br,tb,B0,P(i,:));
    PI(i,:)=x(length(x),1:4); %Final amount of A, B, C and D
    J(i)=PI(i,3)/tb; % Productivity Index (gmoleC/hour)

    % Create graphs if required
    if newrungraphs
        %plot(t,x(:,1),'r-', t, x(:,2),'g--',t,x(:,3),'b-.')
        subplot(2,1,1)
        hold on
        plot(t,x(:,3),'k-')
        xlabel('time (hr)')
        ylabel('Amount C (gmole)')

        subplot(2,1,2)
        hold on
        plot(t,x(:,4),'k-')
        xlabel('time (hr)')
        ylabel('Volume (L)')
        %legend('A','B','C')
        hold off
    end
end

% Initialization of Results to be exported in Excel
xlsResults1={};
xlsResults2={};
xlsResults3={};

% File name to save the data into Excel
if termremoval
    xlsFileName = 'DDoXReaction3Error_removal';
else
    xlsFileName = 'DDoXReaction3Error_noremoval';
end
```

```
figTitle = {'DDoX of Reaction 3 with induced Error';'A + B ==> C ==> D'};
figTitle2 = {'Dynamic Inputs for Reaction 3';'A + B ==> C ==> D'};

% Number of experiments and width of mymodel
    [l,factors]=size(P);

% Induce Error
alphas=[0,0.01,0.02,0.05,0.1];
xopts=zeros(length(alphas)*100-99,factors); %code optimum inputs
xoptsreal=zeros(size(xopts)); %"real-world" optimum inputs
w=0;
p=0;
for i=1:length(alphas)
    if alphas(i)==0
        j=1;
    else
        j=100;
    end

    % Perform Regression and Optimization on 100 replicates
    JMax=zeros(j,1);
    JExact=zeros(j,1);
    JExperi=zeros(j,1);
    for k=1:j
        e=randn(l,1);
        Je=J.*(1+alphas(i)*e);
        p=p+1;

        % Regression model form of the PI Response Surface
        mymodel= [ 0 0;
                   1 0;
                   0 1;
                   1 1;
                   2 0;
                   0 2;
                   2 1;
                   1 2];

        % Perform Regression
        stats=regstats(Je,P,mymodel,{'beta','fstat','tstat','covb'});
        beta=stats.beta; %Model Coefficients
        f = stats.fstat;
        tstats=stats.tstat;
        covb=stats.covb;

        % Remove high p-value terms
        [Y,I]=max(tstats.pval);
        removed=[];
        if termremoval
            while Y>pvalmin
                removed=[removed,I];
                CoefTemp=[];
                [terms,blah]=size(mymodel);
                mymodel=[mymodel(1:I-1,:);mymodel(I+1:terms,:)];
```

```
stats=regstats(Je,P,mymodel,{'beta','rsquare','r','standres','fstat','tstat'}
);
                beta=stats.beta; %Model Coefficients
                f = stats.fstat;
                tstats=stats.tstat;
                [Y,I]=max(tstats.pval);
            end
        end

        % Constrained Optimization of the Response Surface Function
        [xopt,PIMax]=DDoXOptimize(-beta,mymodel); %Negative beta is input so
that a maximum value of J is found
        JMax(k)=-PIMax;
        xopts(p,:)=xopt;
            % Save the "real-world" optimum inputs
            xoptsreal(p,1)=xopt(1);
            xoptsreal(p,2)=xopt(2);

        % Add DDoX Predicted results to be exported to Excel file
        xlsResults2(k+w,2:3)={k,JMax(k)};
        for ii=1:length(xopt)
            xlsResults2(k+w,6+ii)={xoptsreal(p,ii)};
        end

        % Calculate and add DDoX J prediction confidence interval
        X = x2fx(P,mymodel);
        Z = x2fx(xopt,mymodel);
        sigma=(f.sse/(l-length(mymodel(:,1))))^0.5;
        t_val=tinv(1-0.05/2,l-length(mymodel(:,1))-1);
        width=t_val*sigma*(1+Z*(X'*X)^-1*Z')^0.5;
        xlsResults2(k+w,4)={sigma^2};
        xlsResults2(k+w,5)={JMax(k)-width};
        xlsResults2(k+w,6)={JMax(k)+width};

        % Using xopt calculate the Exact Model J
            % Perform simulated experiments
            x0 = [A0, Bt*alp, 0, 0, V0+Bt*alp/B0]; %Initial concentrations
(gmole/L) and initial volume (L)
            [t,x]=ode45(@DDoXReaction3ODE3,[0
tb],x0,[],Br,tb,B0,[xopt(1),xopt(2)]);
        JExact(k)=x(length(x),3)/tb;
        xlsResults2(k+w,12)={JExact(k)};

        % Check to see if this J is within the DDoX confidence interval
        if JExact(k)>=(JMax(k)-width) && JExact(k)<=(JMax(k)+width)
            xlsResults2(k+w,13)={'Yes'};
        else
            xlsResults2(k+w,13)={'No'};
        end

        % Add top conversion from experiment to be exported to Excel file
        A=[Je P];
        [Sorted,I] = sort(A(:,1),'descend');
```

```
        Sorted=A(I,:);
        for ii=1:length(Sorted(1,:))
            xlsResults2(k+w,13+ii)={Sorted(1,ii)};
        end

        % Add empty cells to account for removed model terms
        for ww=1:length(removed)
            www=removed(length(removed)-ww+1);
            tstats.pval=[tstats.pval(1:www-1);-
1;tstats.pval(www:length(tstats.pval))];
            tstats.beta=[tstats.beta(1:www-1);-
1;tstats.beta(www:length(tstats.beta))];
        end

        % Add DDoX Predicted model coefficients and statistics
        xlsResults2(k+w,20)={f.pval};
        for ii=1:length(tstats.pval)
            xlsResults2(k+w,20+ii)={tstats.pval(ii)};
            xlsResults2(k+w,78+ii)={tstats.beta(ii)};
        end
    end

    % Add results to be exported to Excel file
    xlsResults1(i,1:5)={alphas(i),mean(JMax),std(JMax),...
                        mean(JExact),std(JExact)};
    xlsResults2(w+1,1)={alphas(i)};
    w=w+j+1;
end

% Add the design space used information to be exported
for i=1:l
   for j=1:factors
      xlsResults3(i,j)={Preal(i,j)};
   end
end

% Write data to Excel file
if xlsprintdata
    xlswrite(xlsFileName, xlsResults1, 'Sheet1', 'a7');
    xlswrite(xlsFileName, xlsResults2, 'Sheet1', 'g7');
    xlswrite(xlsFileName, xlsResults3, 'Sheet1', 'a17');
end

% Graph the resulting data points
figure
hold on
plot(xopts(1,1),xopts(1,2),'o',...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor','k',...
    'MarkerSize',15)
if length(alphas)>1
    plot(xopts(2:101,1),xopts(2:101,2),'s',...
        'MarkerEdgeColor','r',...
        'MarkerFaceColor','r',...
```

```
        'MarkerSize',10)
    plot(xopts(102:201,1),xopts(102:201,2),'d',...
        'MarkerEdgeColor','g',...
        'MarkerFaceColor','g',...
        'MarkerSize',10)
    plot(xopts(202:301,1),xopts(202:301,2),'^',...
        'MarkerEdgeColor','b',...
        'MarkerFaceColor','b',...
        'MarkerSize',10)
    plot(xopts(302:401,1),xopts(302:401,2),'v',...
        'MarkerEdgeColor','c',...
        'MarkerFaceColor','c',...
        'MarkerSize',10)
end
axis([-1 1 -1 1])
xlabel('a_2')
ylabel('a_3')

% Add edge of constrained space
edg1=[1,0;0,1;-1,0;0,-1;1,0];
plot(edg1(:,1),edg1(:,2),'r-','LineWidth',2)

% Format the font sizes of the figure
set(gca,'FontSize',14);
set(get(gca,'XLabel'),'FontSize',16);
set(get(gca,'YLabel'),'FontSize',16);
set(get(gca,'ZLabel'),'FontSize',16);
set(get(gca,'Title'),'FontSize',18);
% Add legend with the %error data markers
legend('0% error','1% error','2% error','5% error','10%
error','Location','Best')
title(figTitle2)
hold off

% Save figure to file
saveas(gcf,[xlsFileName,'.fig'])


function [xopt,PIMax]=DDoXOptimize(b,mymodel)
% DOX optimization for the batch problem by multiple linear regression.
% Constained optimization
% Reaction 3 DDoX
% by W. Alex Marvin May 2, 2009

% The 4 corner points plus the center point of the design space
P=[fullfact(2*ones(1,2))*2-3; 0,0];
NONLCON=[]; %No nonlinear constraints are required in this classic DoX case
A= [  1  1;
     -1 -1;
      1 -1;
     -1  1];
B=ones(1,length(A(:,1)));
OPTIONS=optimset('GradConstr','on','GradObj','on');
FUN=@(x) DDoXFun(x,b,mymodel);
Aeq=[];
```

```
Beq=[];
LB=[];
UB=[];
xopts=zeros(size(P));
PIMaxs=zeros(length(P),1);
hold on
for ii=1:length(P)
    X0=P(ii,:); %initial inputs guess
    [xopt,PIMax]=fmincon(FUN,X0,A,B,Aeq,Beq,LB,UB,NONLCON,OPTIONS);
    xopts(ii,:)=xopt;
    PIMaxs(ii,1)=PIMax;
end

% Output the minimum xopt value with its related PIMax
[PIMax,I]=min(PIMaxs);
xopt=xopts(I,:);


function dxdt=DDoXReaction3ODE3(t,x,Br,tb,B0,wc)
% Fed Batch Reaction
% A + B --> C --> D
% rate1 = k1*A*B/(1+k2*B^2)
% rate2 = k3*C

% Br = remaining B to be added
% tb = batch time (range from 1 to 2 hr)
% B0 = stock B solution concentration (gmole/L)
% wc = working coefficients [a1,a2] (w(tau) = a1*P1+a2*P2)

% Dimensionless time
tau=t/tb;

% Define the Legandge polynomials
P1= @(tau) -(2*tau-1);
P2= @(tau) 6*tau^2-6*tau+1;

% Coded decision variable w(t)
w=@(tau) wc(1)*P1(tau)+wc(2)*P2(tau);

A = x(1); %(gmole)
B = x(2); %(gmole)
C = x(3); %(gmole)
D = x(4); %(gmole)
V = x(5); %(L)

dxdt = zeros(5,1); % a column vector

% dimentional model paramaters
k1 = 50;    %(L/(gmole*hr))
k2 = 9; %(L/gmole)
k3 = 2; %(1/hr)

% Average flow rate of the B stock solution
u0 = Br/(tb*B0); %(L/hr)
```

```
% Instataneous flow rate of the B stock solution
u =@(tau) u0*(1+w(tau)); %(L/hr)

% dimentional reaction rate
r1 = k1*(A/V)*(B/V)/(1+k2*(B/V)^2); %(gmole/(L*hr))
r2 = k3*(C/V); %(gmole/(L*hr))

dxdt(1)=-r1*V;
dxdt(2)=-r1*V+u(tau)*B0;
dxdt(3)=r1*V-r2*V;
dxdt(4)=r2*V;
dxdt(5)=u(tau);


function [PI,g]=DDoXFun(x,b,mymodel)
% Output the model predicted response
PI=x2fx(x,mymodel)*b;
[n,f]=size(mymodel);
% If required compute the gradient
if nargout>1
    g=zeros(1,f);
    for k=1:f
        modeltemp=mymodel;
        btemp=b;
        for l=1:n
           if modeltemp(l,k)>0
               if modeltemp(l,k)==2
                   btemp(l)=2*btemp(l);
               end
               modeltemp(l,k)=modeltemp(l,k)-1;
           else
               btemp(l)=0;
           end
        end
        g(k)=x2fx(x,modeltemp)*btemp;
    end
end
end
```