UCRL-TR-225940

Methods for Calibration of Prout-Tompkins Kinetics Parameters Using EZM Iteration and GLO

A. P. Wemhoff
A. K. Burnham

November 8, 2006

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory, under Contract No. W-7405-Eng-48.

Approved for public release; further dissemination is unlimited.

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Abstract

This document contains information regarding the standard procedures used to calibrate chemical kinetics parameters for the extended Prout-Tompkins model to match experimental data. Two methods for calibration are mentioned: EZM calibration and GLO calibration. EZM calibration matches kinetics parameters to three data points, while GLO calibration slightly adjusts kinetic parameters to match multiple points. Information is provided regarding the theoretical approach and application procedure for both of these calibration algorithms. It is recommended that for the calibration process, the user begin with EZM calibration to provide a good estimate, and then fine-tune the parameters using GLO. Two examples have been provided to guide the reader through a general calibrating process.

Table of Contents

About the Extended Prout-Tompkins Model 5

Background of the EZM and GLO Calibration Methods 7

Single-Parameter - Single Point Iterative Calibration 8

Coupled erx-zrx Two Point Iterative Calibration 13

EZM Three Point Iterative Calibration 14

GLO Calibration 16

Running the Calibration Codes 18

Performing a Point-by-Point EZM Calibration 23

Performing a GLO Calibration 24

Running a Set of Sequential Runs with the Same Kinetic Parameters 25

Examples 26

Example 1: Two-Point Calibration 26

Example 2: Three-Point Calibration 29

References 32

About the Extended Prout-Tompkins Model

A simple, universal, and functionally accurate kinetics model of the thermal decomposition of various explosives would save computational resources for the incorporation of non-traditional explosives into large-scale cookoff simulations. Wemhoff and Burnham [1] have explored substituting a sequential reaction process by a single reaction model that incorporates aspects of nucleation and propagation: the extended Prout-Tompkins model [2],

$$\frac{dx}{dt} = -kx^n(1-qx)^m = -k(1-qx) \tag{1}$$

where x is the fraction remaining of reactant, t is time, and

$$k(T) = A\exp\left(-\frac{E}{RT}\right) \tag{2}$$

where A is the frequency factor in units of reciprocal time and E is the activation energy in units of energy per mole. In this document, for clarity we introduce the parameter p,

$$p = -\log_{10}(1-q) \tag{3a}$$

which may also be solved for q as

$$q = 1-10^{-p} \tag{3b}$$

which contains the property that if p contains no nonzero values after the decimal point, then it is identical to the number of 9's after the decimal point for q. For example, for q = 0.9999, p = 4.0.

In order to use the above kinetics model, Wemhoff and Burnham [1] explored the effects of adjusting the various parameters on the time to explosion for the ODTX experiment. Therefore, several test runs of RDX were performed to determine the effect of varying these parameters. In these runs, the parameters E/R and A were maintained at 20000 K and 1×10^{15} per sec, respectively, except in a single case where A was adjusted to match the highest temperature data point between curves for n = 0, m = 1 and n = 0, m = 0. The two extreme values of p were used in Figures 1a and 1b (2 and 9, respectively), where any value of q less than 2 is unphysical per the Prout-Tompkins approximation (see [1]), while any value higher than 9 is read as q = 1 into ALE3D, which causes no reaction to proceed per Eq. (1). The values of n used were 0.0 and 1.0, while the values of m used were 0.0, 0.5 and 1.0. These curves suggest the following:

- Increasing the parameter p tends to flatten out the ODTX trend and generally increases the time to explosion (i.e. it shifts the curve upwards).

- Increasing the parameter *m* also tends to flatten out the curve and shifts it upward in a manner similar to increasing *p*, although the value of *m* has a more pronounced effect on the location of the bend in the curve.
- Increasing the parameter *n* has very little effect on the calculation except near the critical temperature.

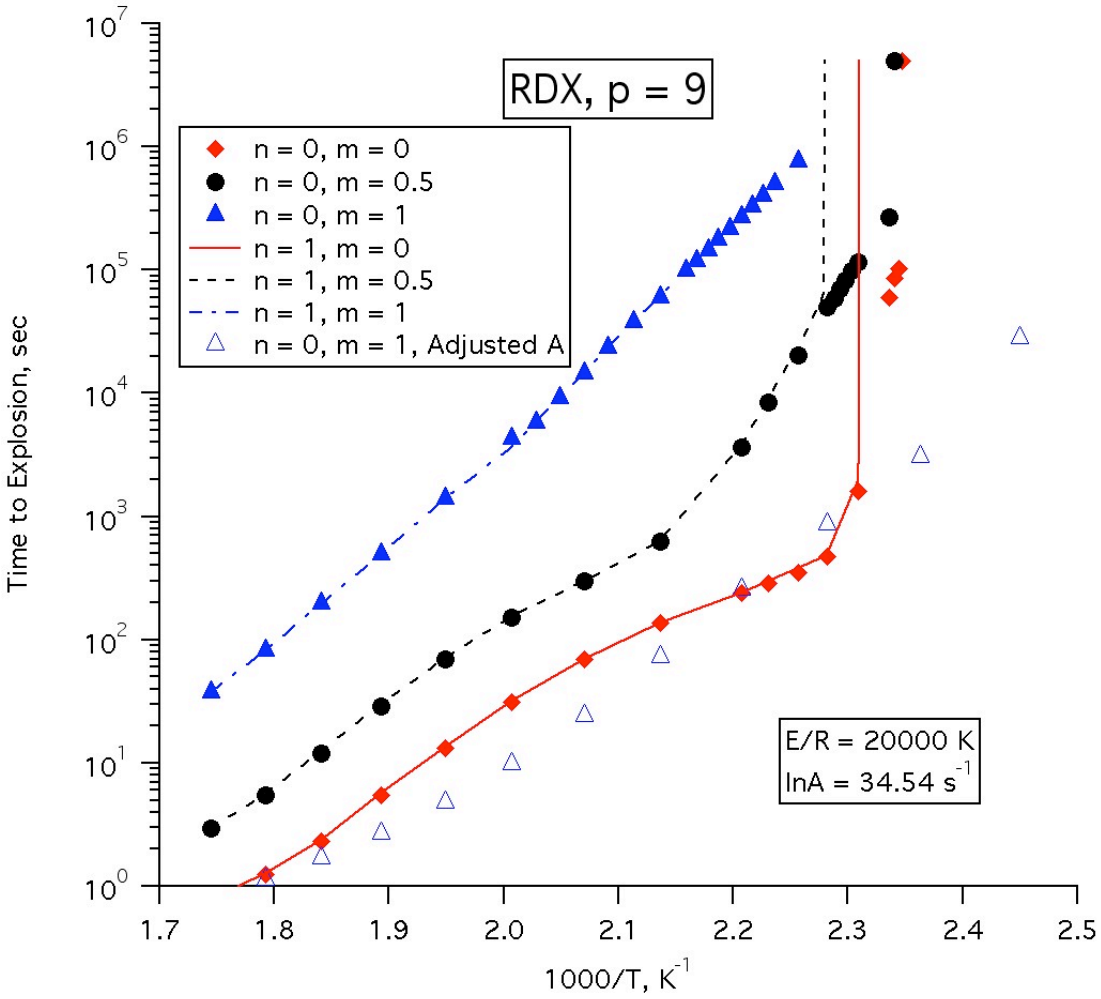

FIGURE 1a. Simulated ODTX trends for RDX for $p = 9$. Value of adjusted frequency factor for open triangles is 1.04×10^{18} per sec.

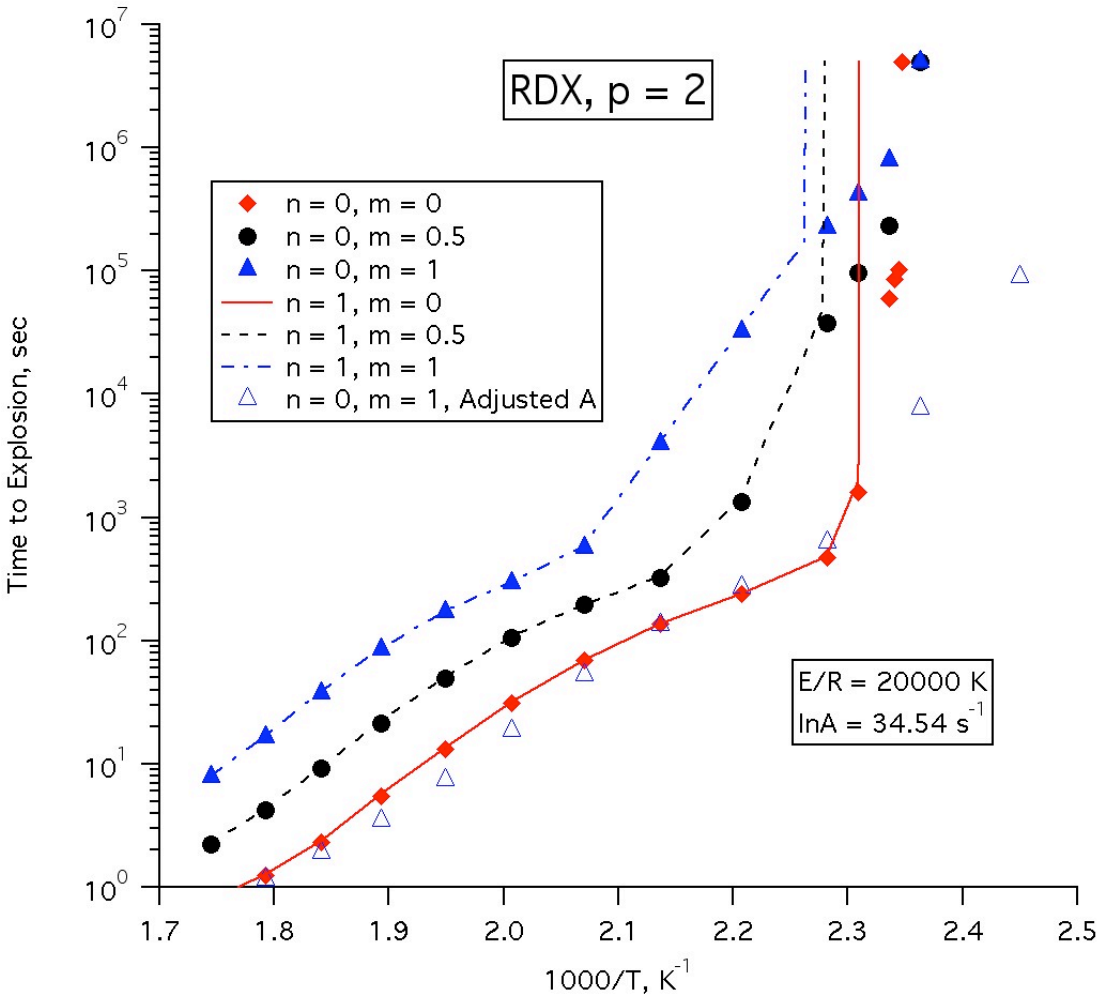

FIGURE 1b. Simulated ODTX trends for RDX for $p = 2$. Value of adjusted frequency factor for open triangles is 5.70×10^{16} per sec.

Background of The EZM and GLO Calibration Methods

The general knowledge of the characteristics of varying these parameters allows for the choice of calibration techniques for a given explosive. For example, Figure 1a shows that the curvature of the ODTX curve reduces as p increases. In addition, Figure 1b shows that the choice of m has a large influence on the location of the bend in the ODTX curve. Therefore, knowledge of the calibration data allows for two general approaches for calibration:

- If the experimental data appear in an approximately straight line, then set the value of p to its highest allowable value (generally 9) after the decimal point, $n = 0$, and $m = 1$. A and E are the primary parameters optimized. The initial use of $n = 0$ is preferred for numerical stability.

- If the experimental data contain a large increase in explosion times at low temperatures, then set the value of p to a lower number (generally 3) and adjust the values of A, E, and m accordingly to create the bend in the simulated curve.

The approach for each explosive generally falls under one of the two above strategies. The dependence on n is sufficiently weak that it may be optimized in a final fine-tuning to match low-temperature data if necessary. However, ODTX data is usually insufficient to constrain n, and its value is probably best determined by independent evidence. Thermal analysis data supports n values closer to unity. Better guidance is a matter of current research.

The first step in calibration is to determine representative data points for use in the simulations. The LLNL Explosives Guide [3] provides experimental ODTX results for a wide variety of explosives and temperatures. Generally, one or several of these data points are used for a given calibration technique, and the choice of data points should provide a reasonable approximation of the explosion time-temperature curve.

There are several approaches for calibrating the parameters in Eq. (1) to match ODTX data. These calibration techniques are now described, ranging from simplest to most complex. The increase in complexity in the iterative calibration techniques is meant to follow an increase in usefulness of the techniques for attaining accurate parameters. In ALE3D, the Prout-Tompkins reaction is generally input as

$$k\left(x_s,T\right) = \exp\left(zrx - \frac{erx}{RT}\right) x^n \left(1 - qx\right)^m \tag{4}$$

where $erx = E$ in cal/mol, and $zrx = \ln\left(10^{-6} A\left[\mathrm{s}^{-1}\right]\right)$. It should be noted that the parameter $rhopow = 1 - m - n$ is used to convert the Prout-Tompkins reaction from the default mass concentration basis to a mass fraction basis in ALE3D. The only reaction considered in this study is the direct decomposition of solid material to gaseous products, and therefore melting was not taken into account. Therefore, the parameters to be calibrated are zrx, erx, m, n, and q.

Single Parameter - Single Point Iterative Calibration

This algorithm allows the user to calibrate the parameters in ALE3D by adjusting a single kinetics parameter to match results for a single datum. In this simple program, the user inputs the following variables:

- parameter to be optimized
- goal time in seconds
- maximum number of iterations
- minimum and maximum parameter value
- run temperature
- convergence tolerance
- parameter adjustment algorithm

The code first checks that the goal may be reached by determining explosion times for the minimum and maximum parameter values. If the goal lies between these calculated explosion times, then calibration is possible. An example is shown schematically in Figure 2, where the time-to-explosion t increases as the parameter m increases, and the time-to-explosion goal t_{goal} is found to exist for a parameter value in the range $m_{min} < m < m_{max}$, assuming that there exists only one value of t for a given m. Note that the descriptors *min* and *max* refer to the parameter m only; i.e. the value of t_{max} may be lower than t_{min} if the general slope of the curve is negative.

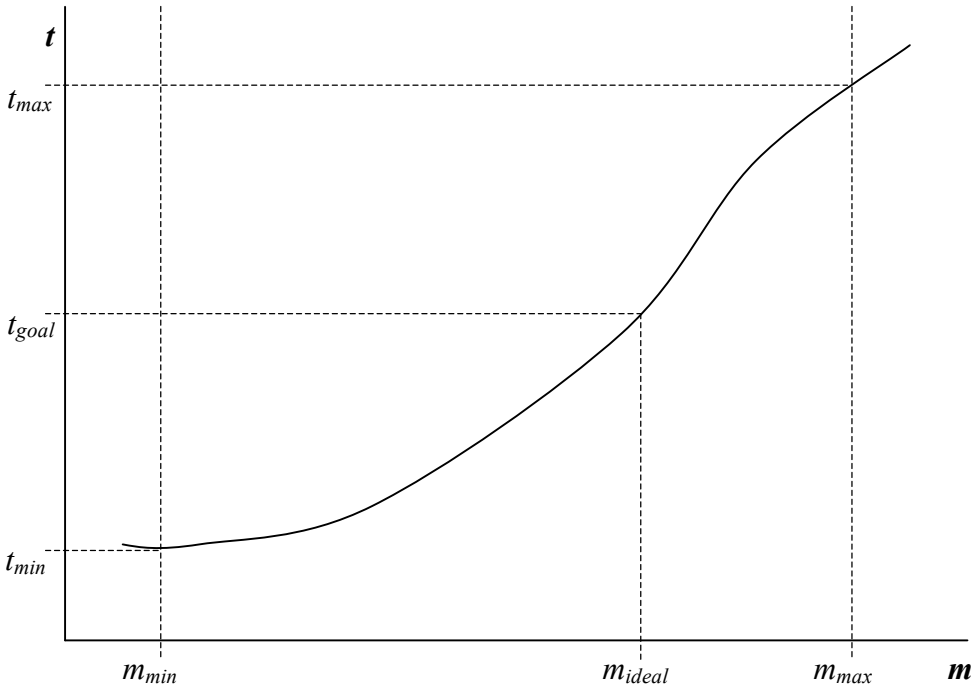

FIGURE 2. Schematic of example relationship between m and t.

The code attempts to calibrate the parameter in two ways. These procedures are repeated until either the maximum number of iterations or the convergence criterion has been met. The linear interpolation algorithm is the default method for all iterative calibration techniques.

- *Simple bracketing*: if none or only one of the calculated times-to-explosion are known for the edges of the bracketing window, then the new parameter value is at the center of the bracketing window as shown in Figure 3, and mathematically by

$$m_{new} = \frac{1}{2}\left(m_{\min} + m_{\max}\right) \tag{11}$$

For each iteration, the algorithm also replaces the value of either m_{min} or m_{max} based upon the calculated value of t, which reduces the bracketing window by one-half during each iteration.

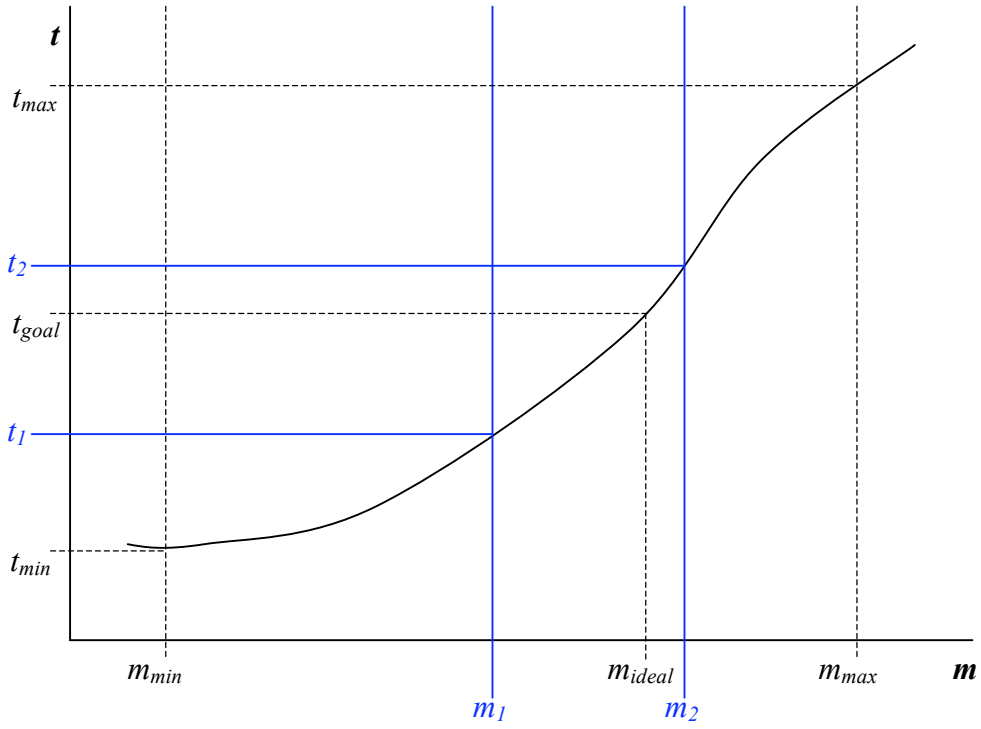

FIGURE 3. Simple bracketing algorithm schematic.

- *Linear interpolation*: if both calculated times-to-explosion are known at the edges of the bracketing window, then the new parameter value is found using linear interpolation as shown in Figure 4. The updated value is found by

$$m_{new} = m_{\min} + \left(m_{\max} - m_{\min}\right)\frac{\left(t_{goal} - t_{\min}\right)}{\left(t_{\max} - t_{\min}\right)} \tag{12}$$

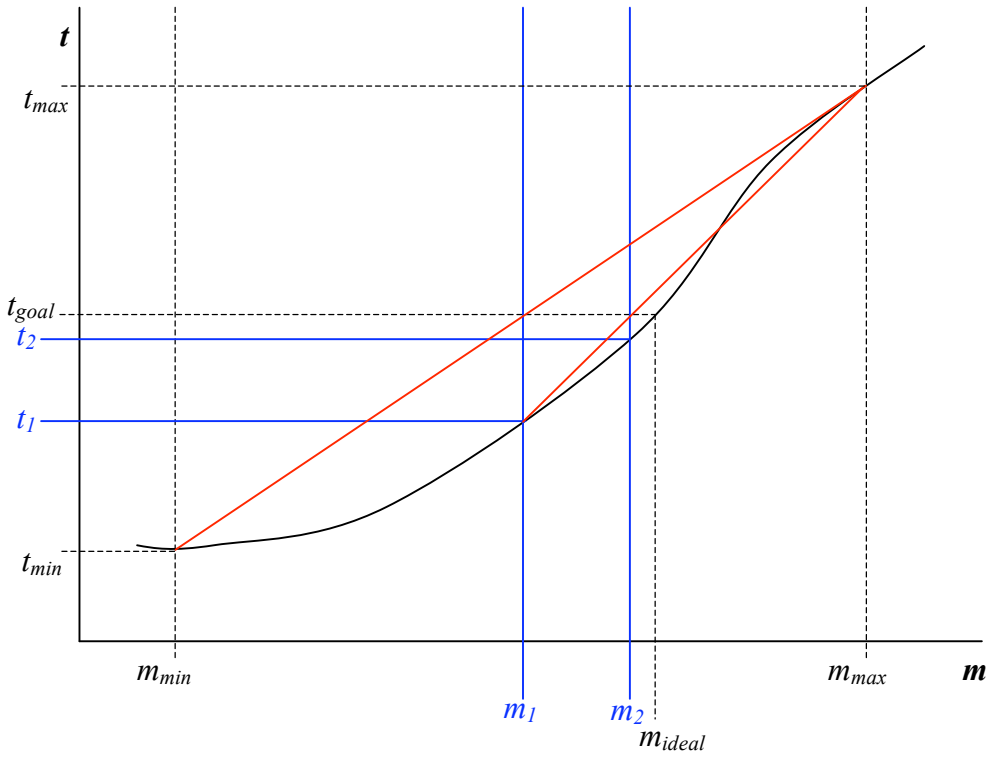

FIGURE 4. Linear interpolation method schematic.

- *Shifted linear interpolation*: this algorithm provides a modification to the standard linear interpolation algorithm to take curvature effects into account. An example of this is shown in Figure 5, where the large curvature of the time curve shifts the value of m_2 from m_{2L} to m_{2P}. This algorithm is only implemented when the calculated time is far from the goal and when no shifting had taken place the previous time step. An applied relaxation factor continually updates after each shifting step to increase accuracy. Table 1 shows that studies of this algorithm have shown significant reduction in the required number of iterations compared to where no shifting was applied when the curvature of the dependent variable curve is large.

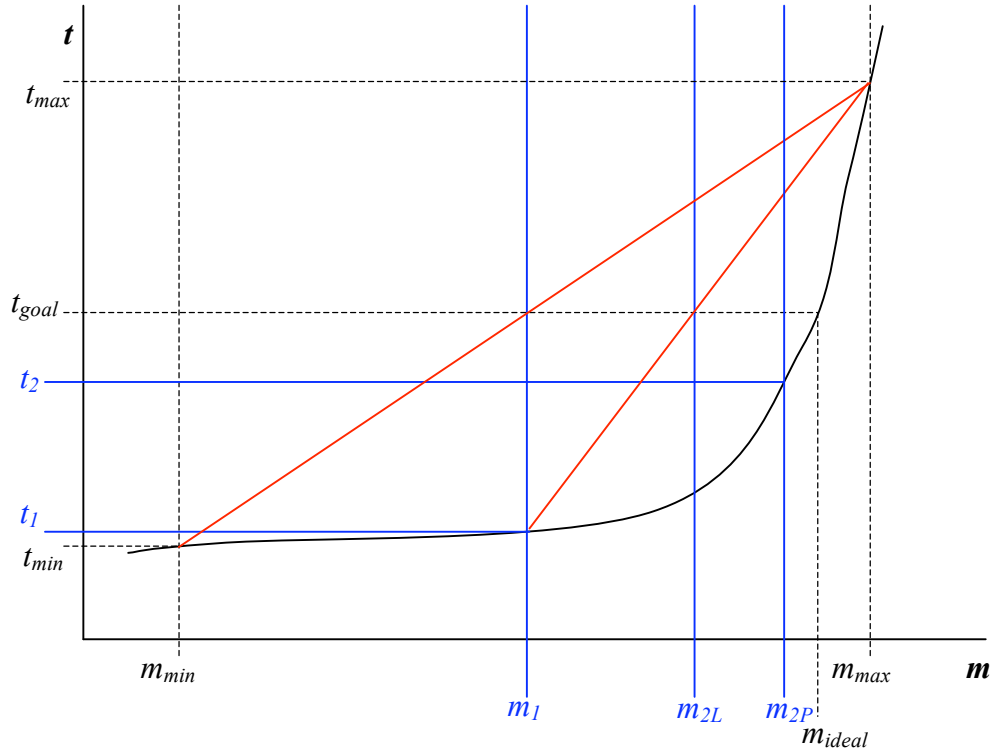

FIGURE 5. Shifted linear interpolation method schematic.

TABLE 1. Comparison of convergence efficiency from application of shifting algorithm. The range for all equations is $0.1<x<5.0$, the goal is 15.0, and the convergence tolerance is 1%.

Equation $t = f(m)$	**Steps Required for Convergence**	
	Linear Interpolation Without Shifting	**Linear Interpolation with Shifting**
$t = \exp(m)$	23	6
$t = m^2$	5	5
$t = m^3$	15	7
$t = 1000\exp(-m)$	67	8
$t = 100\ln(1+m)$	5	4

Coupled *erx*-*zrx* Two Point Iterative Calibration

The single parameter-single point iterative calibration tool assumes no coupling between the parameters. However, the compensation relationship between the parameters *erx* and *zrx* allow for coupling by using two data points. The rationale for this may be seen in Equation (4), where the rate of reaction may be expressed as

$$k(x,T) = k_c(T)x^n(1-qx)^m \tag{5}$$

where

$$k_c(T) = \exp\left(zrx - \frac{erx}{RT}\right) \tag{6}$$

This algorithm performs two steps. First, any parameter (although generally either *erx* or *zrx* is chosen) is adjusted for a single temperature T_c using the aforementioned single parameter-single point iterative algorithm. Then, the value of k_c is determined for this temperature using Eq. (6). Then, the single parameter-single point algorithm is repeated at a different temperature. Preservation of $k_c(T_c)$ requires that

$$erx = RT_c(zrx - \ln k_c) \tag{7}$$

The resultant adjusted values of *erx* and *zrx* provide times to explosion that match two calibration points within the convergence tolerance. Figure 6 depicts what an example simulated curve would look like using this calibration method.

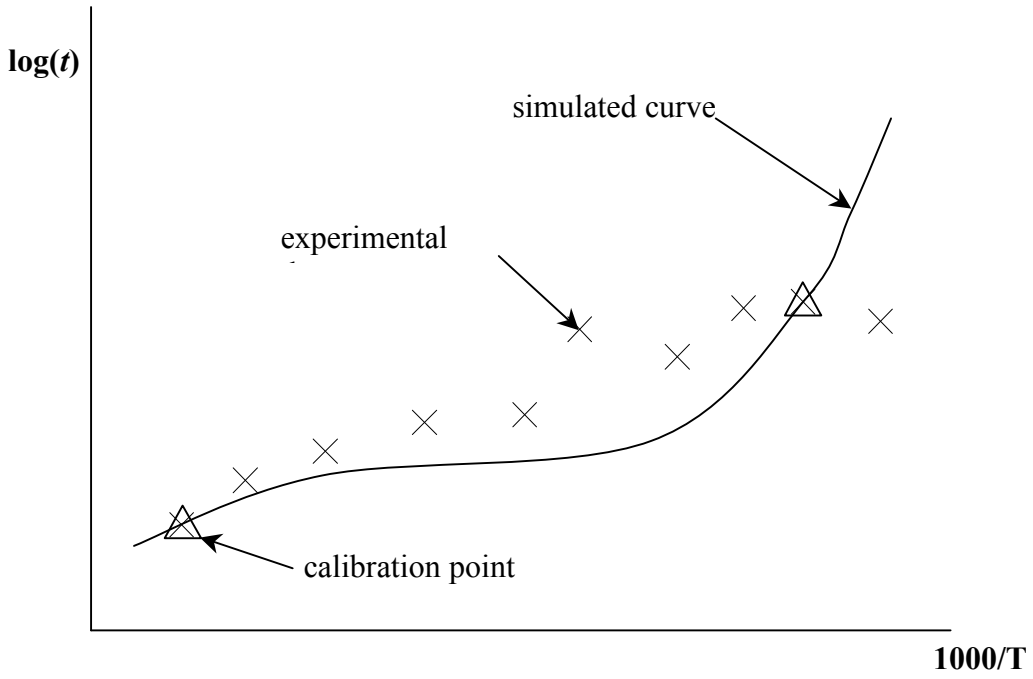

FIGURE 6. Schematic of resultant calibrated curve using coupled *erx-zrx* two-point calibration.

EZM Three Point Iterative Calibration

The EZM (*erx-zrx-m*) three point iterative calibration algorithm is an extension of the coupled *erx-zrx* two point iterative calibration to allow for additional adjustment of *m* for a third data point. Figure 7 provides a depiction of how this algorithm works. At each step, the value of *m* is iterated based on a calculated time for a third data point, where the parameters *erx* and *zrx* are previously calibrated for two other points on the curve. The resultant simulated curve passes through three calibration points as shown in Figure 8. Although this technique is powerful, successful completion of the calibrator calls for accurate initial values of *erx*, *n*, and *q*, and correct bounding limits on *zrx* and *m*.

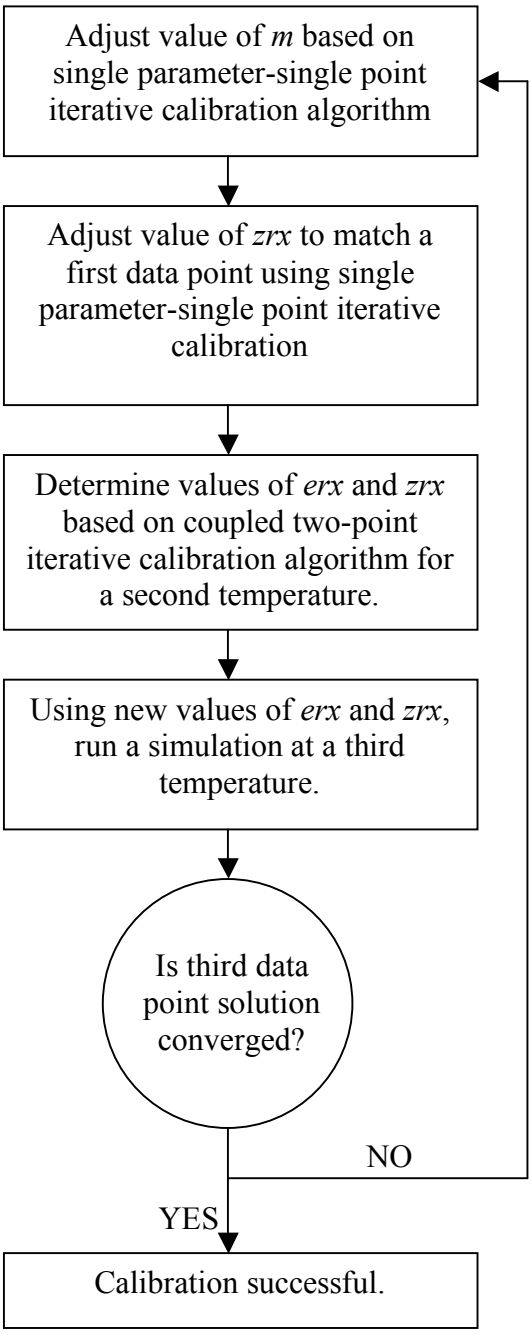

FIGURE 7. EZM iterative calibration algorithm.

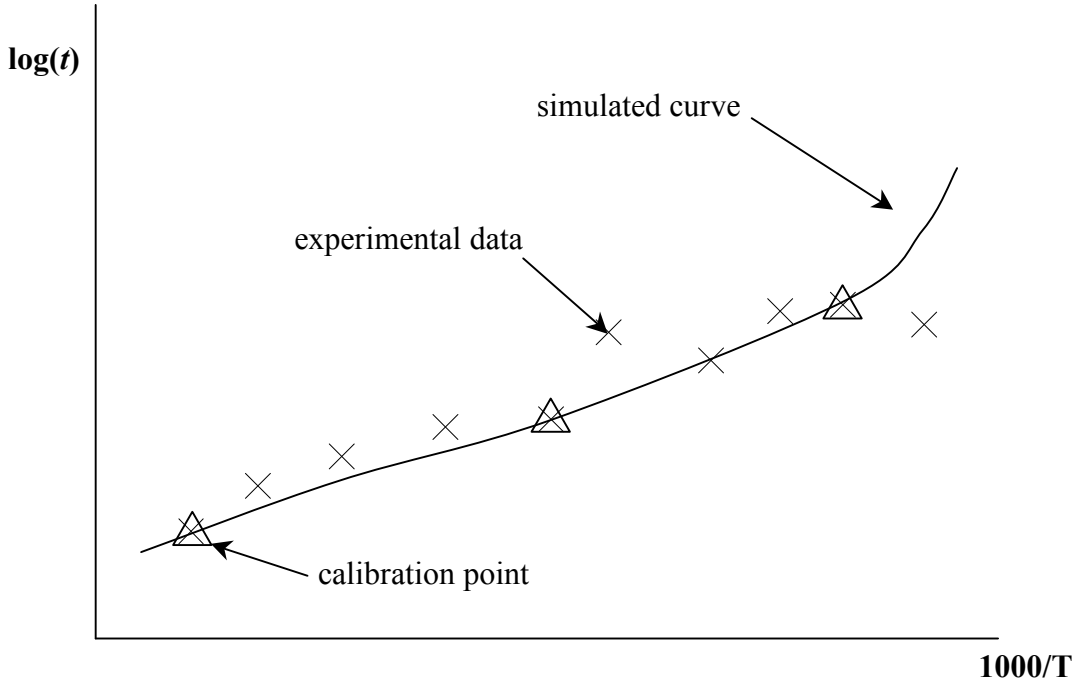

FIGURE 8. Schematic of resultant calibrated curve using EZM calibration.

GLO Calibration

Although the EZM calibration tool provides an accurate means to calibrate parameters to match experimental data, the algorithm is limited to only three calibration points and to the parameters *erx*, *zrx*, and *m*. Use of LLNL's Global Local Optimizer (GLO) code allows for calibration of multiple parameters for multiple calibration points. The GLO code allows for two choices for optimization: global and local [4]. The global optimization tool is generally used for parametric studies, while the local optimization tool searches for local minima using the method of steepest descents. Since kinetic parameter calibration requires a local optimization, the local optimization tool was primarily used in this study. The input to the code is a bracketing window for all parameters (similar to that shown in Figure 2), and initial values for each. Another input is the Figure-of-Merit (FOM) evaluation, which was defined as

$$FOM = \sum_{i=1}^{N_p} \left[\ln\left(t_{calc} / t_{goal} \right) \right]_i^2 \tag{8}$$

where N_p data points were used. For each GLO iteration, the parameters were placed in a Perl script, which in turn spawned an ALE3D run for each of the data points, as shown in Figure 9. The results from all ALE3D runs were stored in a text file, which was then read

for post-simulation determination of the FOM. This FOM was then fed back into GLO for consideration of the adjustment of parameters.

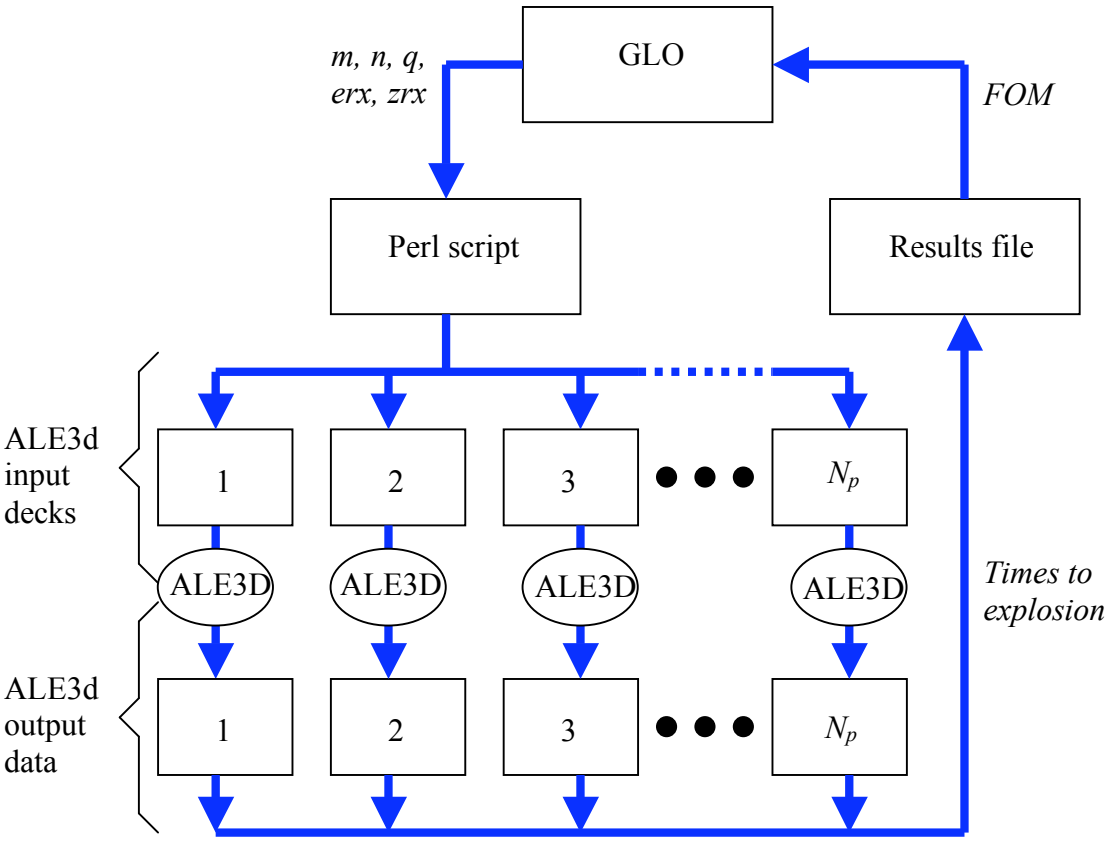

FIGURE 9. Data flow schematic for a GLO run.

The code adjusts the parameters by first determining the gradients along the FOM surface, and then taking a Newton step using the method of steepest descents. If the result of the Newton step yields a FOM larger than at the previous location, then a 1-d minimization subroutine is called to determine the lowest FOM along the 1-d slice between the points before and after the Newton step. Figure 10 shows a typical history of the FOM value for a GLO run where two parameters are calibrated.

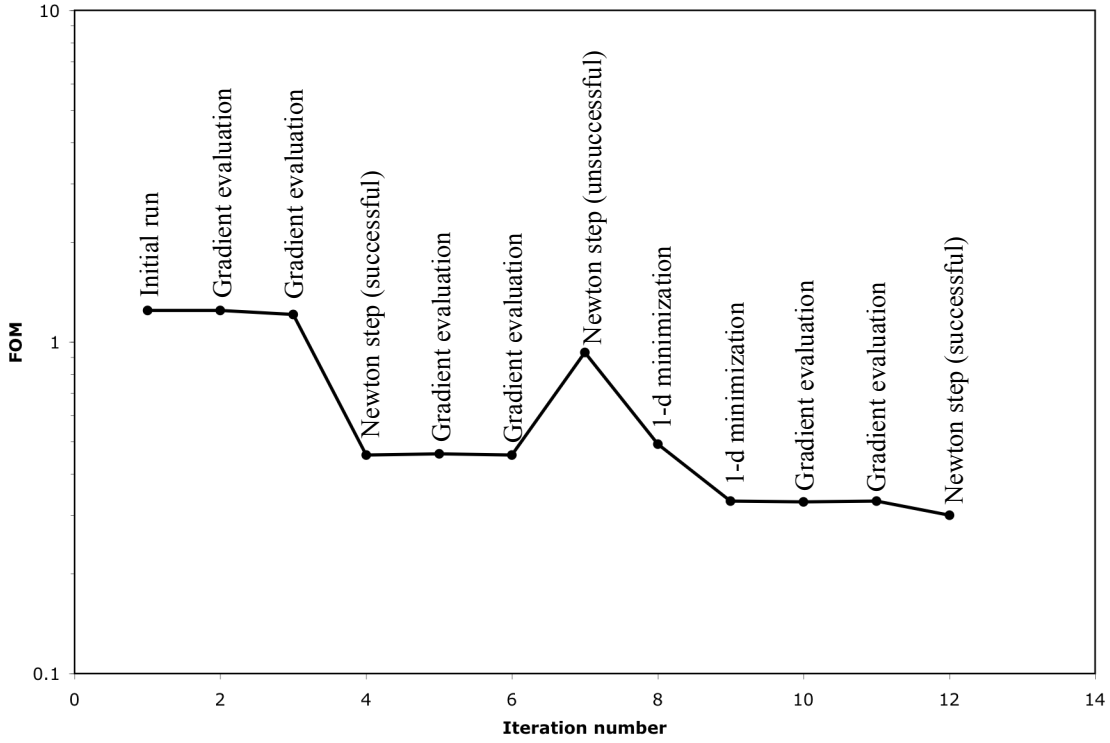

FIGURE 10. Example Figure of Merit adjustment history for a simulation where two parameters are calibrated.

Running the Calibration Codes

When initializing a calibration process, the first step is to make sure that the working directory contains all the necessary files. These are:

1. `PT_inputmake.pl`: creates (and has the option to run) the EZM calibrator script.
2. `b-ezm_calibrator.pl`: base EZM calibrator file (not used directly).
3. `auto_optimizer.pl`: runs either a one-parameter optimization or a coupled *erx-zrx* calibration.
4. `autovary_odtx.pl`: runs an ODTX or STEX simulation at a specified temperature or ramp rate.
5. `b-odtx-lx10-100.in`: base ALE3D input deck (not used directly).
6. `b-lx10-kin_bd_v01.in`: base ALE3D kinetics input deck (not used directly).
7. `b-lx10-mtl_bd_v01.in`: base ALE3D material input deck (not used directly).
8. `cleaner`: list of UNIX commands to remove unnecessary files after a calibration run.
9. `sphh.sami`: Mesh used in ODTX simulations.
10. `glomake.pl`: Creates the GLO input file for fine-tuning of kinetic parameters.
11. `auto_cleanup.pl`: Cleans up directories created during a GLO run.

Several other files will be created during the calibration process. Figure 11 below sketches out how the codes create new files:

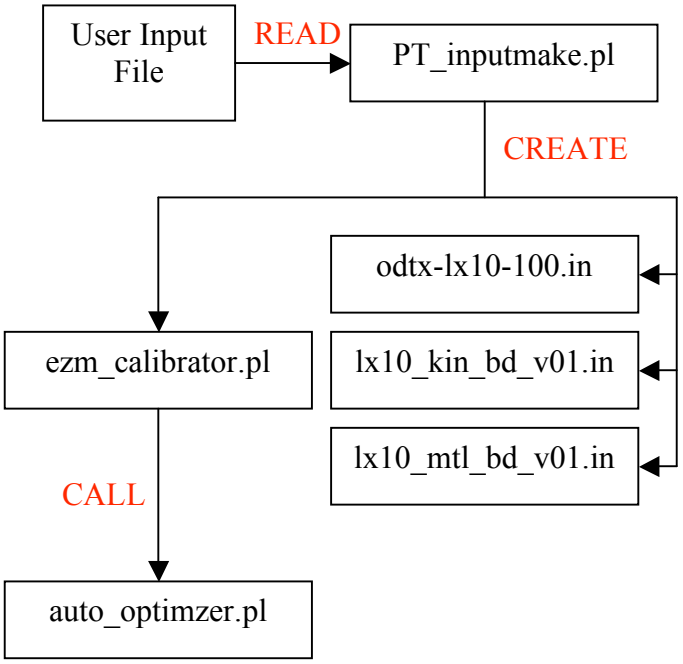

FIGURE 11. Files created during the calibration process.

The second step is to create the appropriate input decks using `PT_inputmake.pl`. This script reads in the general parameters of the simulation, and creates the appropriate EZM script setup along with the ALE3D input decks. Table 2 lists the parameters to be input to `PT_inputmake.pl`. The rules for this input deck are as follows:

- Lines with parameter names must include at least one space between the name and value.
- The parameter name must appear prior to the parameter value.
- ODTX data must include at least one space between the name and ODTX temperature, and at least one space between the ODTX temperature and ODTX time.
- Commented lines contain an asterisk "*" in the first column.
- Blank lines are allowed.
- Blank spaces before variable names are allowed but not required.

To run the script, type at the prompt

```
perl PT_inputmake.pl
```

and supply the appropriate command-line flags (default values in parentheses, input line flags in brackets):

- Input file name (input.txt)[**-i**]
- Run the EZM calibrator automatically after creating the ALE3D input deck [**-r**]

- Obtain help info [**-h**]

TABLE 2. Input variables to `PT_inputmake.pl`.

Parameter	Default Value(s)	Description
General Material Properties		
matname	gen_mat	Material Name
ref_density	1.0	Reference density, g/cc
solid_cv	1.e-5	Solid heat capacity at constant volume, Mbar-cc/(g-K)
solid_cond	1.e-14	Solid thermal conductivity, Mbar-cc/(cm-us-K)
prod_cv	1.e-5	Products heat capacity at constant volume, Mbar-cc/(g-K)
prod_cond	5.e-15	Products thermal conductivity, Mbar-cc/(cm-us-K)
prod_energy	1000	Energy released by reaction in cal/g
prod_coeff	0.3	Products gamma-law gas coefficient
Prout-Tompkins Calibration Parameters		
pt_p	9	Prout-Tompkins parameter $p = -\log_{10}(1-q)$
pt_n	0	Prout-Tomkins parameter n
itmax_m	20	Maximum number of iterations over the Prout-Tompkins parameter m
itmax_z	20	Maximum number of iterations over the Prout-Tompkins parameter zrx
tol_m	0.01	Convergence tolerance for m
tol_z	0.01	Convergence tolerance for z
pt_mmin	0	Minimum value of m
pt_mmax	1	Maximum value of m
pt_zmin	5	Minimum value of zrx
pt_zmax	30	Maximum value of zrx
ignore_warn	0	Ignore warnings from ALE3D? = 1 for yes
verbose	0	Verbose output from the EZM calibrator? = 1 for yes
convquit	0	Quit EZM calibrator if there's no convergence? = 1 for yes
ALE3D Run Parameters		
gen3dexe	/usr/apps/ale3d/bin/gen3d	Path for GEN3D executable
ale3dexe	/usr/apps/ale3d/bin/ale3d	Path for ALE3D executable
use_srun	0	Use SLURM (srun) to run ALE3D? = 1 for yes

TABLE 2. Input variables to `PT_inputmake.pl` (continued).

Parameter	Default Value(s)	Description
ODTX Data		
odtx_pts	2	Number of ODTX points (2 or 3)
odtx_1	0 0	Parameter 1: ODTX temperature (°C) for point 1 Parameter 2: ODTX time (sec) for point 1
odtx_2	0 0	Parameter 1: ODTX temperature (°C) for point 2 Parameter 2: ODTX time (sec) for point 2
odtx_3	0 0	Parameter 1: ODTX temperature (°C) for point 3 Parameter 2: ODTX time (sec) for point 3

Note that if odtx_pts is equal to 2, then the following occurs automatically:

- Any input for `odtx_3` is ignored.
- The parameter `pt_p` is set to 9.
- The parameter `itmax_m` is set to 1.
- The parameter `tol_m` is set to 1.
- The parameters `pt_mmin` and `pt_mmax` are set to 1.
- The parameter `pt_n` is set to 0.

A sample input deck to `PT_inputmake.pl` is seen below.

```
* This is a sample input deck for the PT_inputmake.pl.
* All entries here are default values and should be entered in
* B-division units, except the product energy in cal/g.  Some of
* these parameters are the same as the default and thus may be
* ommitted.

  matname mymat
  ref_density 1.61
  solid_cv 1.3e-5
  solid_cond 9.5e-14
  prod_energy 1108
  prod_cv 1.1e-5
  prod_cond 7.9e-15
  prod_coef 0.279

* These are used for determining the calibration procedure.
  pt_p 3
  itmax_m 20
  itmax_z 26
  tol_m 0.2
  tol_z 0.2
  pt_mmin 0
  pt_mmax 1
  pt_n 8
  pt_zmin 1
  pt_zmax 50
  ignore_warn 1
  verbose 1
  convquit 0

* These commands are for the path of the ale3d executable &
* whether to use SLURM (srun)
  gen3dexe /usr/apps/ale3d/bin/gen3d
  ale3dexe /usr/apps/ale3d/bin/ale3d
  use_srun 0

* ODTX values should be listed as a temperature (C) and a time
* (sec).  The paramter odtx_pts describes the # of pts to be fit
* to.
  odtx_pts 3
  odtx_1 255 19.8
  odtx_2 165 56916
  odtx_3 190 2828.2
```

Two example calibrations are provided later in this report. The code that actually performs the calibration is `ezm_calibrator.pl`. The first few lines of the code contain the input parameters provided by `PT_inputmake.pl`. The EZM calibration is initiated by typing

```
perl ezm_calibrator.pl
```

or by adding the **-r** flag to the command line for `PT_inputmake.pl`.

If any calibration fails, then a more involved approach is needed. Any given calibration run may not be successful for a variety of reasons, including:

- The input ranges of *zrx* or *m* are invalid.
- More iteration steps may be needed for convergence (if `convquit` = 1).
- An ALE3D run may list warnings in chemical or thermal convergence, causing the calibrator to stop (if `ignore_warn` = 0).

The following section describes how to perform the calibration with more control by using the "built-in" perlscripts.

Performing a Point-by-Point EZM Calibration

The `ezm_calibrator.pl` code acts as a manager of the perlscript `auto_optimizer.pl`. The latter code calibrates the kinetics for each individual point. To run `auto_optimizer.pl`, type at the command prompt

```
perl auto_optimizer.pl
```

and supply the appropriate command-line flags (default values in parentheses, input line flags in brackets):

- Reaction name (pt_solid_to_prod)[**-r**]
- Name of main ALE3D input deck (odtx-lx10-100.in) [**-mainf**]
- Name of ALE3D kinetics input deck (lx10_kin_bd_v01.in) [**-kinf**]
- Parameter to be optimized (zrx)[**-v**]
- Temperature (205.1) [**-t**]
- Goal time in seconds (23.4) [**-g**]
- Maximum number of iterations (10)[**-i**]
- Minimum parameter value (5)[**-low**]
- Maximum parameter value (50) [**-high**]
- Explosion time corresponding to minimum parameter value (unknown) [**-lowt**]
- Explosion time corresponding to maximum parameter value (unknown) [**-hight**]
- Convergence tolerance (0.01) [**-c**]
- Pinned temperature in deg C for coupled *erx-zrx* calibration [**-temp_pin**]
- Use simple bracketing instead of linear interpolation [**-simple**]
- Turn off shifting in linear interpolation [**-noshift**]
- Maximum ALE3D step size in microseconds (10^5) [**-stepsize**]
- Apply a time limit to the ALE3D runs (2× the goal time) [**-time_limit**]
- Ignore warning messages option [**-ignore_warn**]
- Use srun (SLURM) for Linux operating systems [**-use_srun**]
- Specify GEN3D path (/usr/apps/ale3d/bin/gen3d) [**-gen3dexe**]
- Specify ALE3D path (/usr/apps/ale3d/bin/ale3d) [**-ale3dexe**]

Generally, the only command line flags commonly used for a single-parameter single-point calibration are **-r**, **-v**, **-t**, **-g**, **-low**, **-high**, and **-stepsize**. The purpose of the **-lowt** and **-hight** flags is to allow the user to "restart" the calibration procedure when the

optimal value for the parameter lies outside the input range. Note that the **-d** flag provides the current kinetic parameters drawn from the kinetics file, and the **-h** flag provides help regarding the various input flags.

Example 2 below will show how to perform this calibration.

Running `auto_optimizer.pl` generates the following output files in addition to the typical ALE3D output files:

- `g-`*`mainfile`*`-i.in`, where *`mainfile`* is the main ALE3D input file, and *`i`* is the iteration number: the modified input file for iteration *i*.
- `optimized.res`: parameter value and corresponding explosion time for each iteration
- `warnings.out`: any warning messages from ALE3D that terminates the script (deleted if empty at the end of the run)

Performing a GLO Calibration

A GLO input generator perlscript `glomake.pl` has been developed as an aid to the user. This perlscript requires a base ALE3D input deck divided into three files: a main file, a materials file, and a kinetics file. The generator copies and modifies these files into the GLO input deck. The following is needed when running `glomake.pl` (default values in parentheses, and input line flags in brackets):

- Name of reaction (pt_solid_to_prod) [**-r**]
- Name of main ALE3D input deck (odtx-lx10-100.in) [**-mainf**]
- Name of ALE3D kinetics input deck (lx10_kin_bd_v01.in) [**-kinf**]
- Name of ALE3D materials input deck (lx10_kin_bd_v01.in) [**-mtlf**]
- Name of temperature data points for calibration (datapoints.txt) [**-datf**]
- Name of GLO input deck to be created (out.gcf) [**-o**]
- Maximum number of GLO iterations (50) [**-max**]
- Number of cpu's to be used in a GLO run (1) [**-ncpu**]
- Use srun (SLURM) for Linux operating systems [**-use_srun**]
- Specify GEN3D path (/usr/apps/ale3d/bin/gen3d) [**-gen3dexe**]
- Specify ALE3D path (/usr/apps/ale3d/bin/ale3d) [**-ale3dexe**]

Multiple reactions and materials are allowed in the materials and kinetics input decks. All changes in the kinetics input decks are applied to the specified reaction. In the data points input, each point is read in as two columns: temperature and goal time. Commented lines contain an asterisk (*). The same input rules for the `PT_inputmake.pl` input file apply for the input to `glomake.pl` as well. An example valid data points input file `datapoints.txt` is shown below:

```
* temp goal
300.0   12.1
285.0   24.3
260.3   65.9
250.1   571.6
240.6   2842.0
```

Upon completion of the deck, GLO may be run interactively using

```
glo out.gcf
```

where here `out.gcf` is the name of the GLO input deck, and glo is aliased to the executable for a given machine (see the GLO user manual [4] for details). The GLO input deck name may be set using the **-o** flag in `glomake.pl`. This code is also commonly run in the background via

```
glo out.gcf >& out.out < /dev/null &
```

Similar command-line arguments are used for GLO restarts, and further information is available in the GLO user manual [4].

Running a Set of Sequential Runs with the Same Kinetic Parameters

In many cases, the calibrated parameters are based on a fraction of the total available ODTX data for a given explosive. Therefore, the user may want to compare predictions from the calibrated model to the remaining unused ODTX data points, or verify the accuracy of the calibrated parameters to used ODTX data points. The script `autovary_odtx.pl` was created for this purpose. This simple script allows the user to run a series of ALE3D simulations at different temperatures (or ramped boundary conditions). Command line flags for this script are:

- Parameter to be adjusted: (temp) or ramp [**-v**]
- Problem geometry: (0.5sph), 1sph, 2sph, 0.02cy, 2cy, 0.02cy2D, 2cy2D, SITI [**-g**]
- Number of runs: (1) [**-imax**]
- Initial parameter value (1) [**-init**]
- Sequential change in parameter value (1) [**-delta**]
- Maximum ALE3D step size in microseconds (10^5) [**-stepsize**]
- Add delta instead of subtract delta for each subsequent run [**-add**]
- Keep all ALE3D files after run [**-noclean**]
- Ignore warning messages after run [**-ignorewarn**]
- Specify a time limit in seconds [**-time_limit**]
- Use srun (SLURM) for Linux operating systems [**-use_srun**]
- Specify GEN3D path (/usr/apps/ale3d/bin/gen3d) [**-gen3dexe**]
- Specify ALE3D path (/usr/apps/ale3d/bin/ale3d) [**-ale3dexe**]
- Use a list of temperatures instead of a fixed increment [**-list**]

The **-list** flag overrides any of the fixed-increment flags **(-imax**, **-init**, and **-delta**). The user specifies the name of the file with the temperatures and goal values (exactly the same format as the data points file mentioned for `glomake.pl` above). For example, if the user types the command:

```
perl autovary_odtx -init 300 -delta 5 -imax 2
```

the script will provide 3 ALE3D simulations: at 300 C, 295 C, and 290 C. If the user types the same command, except adds the **-list** flag:

```
perl autovary_odtx -init 300 -delta 5 -imax 2 -list
datapoints.txt
```

where `datapoints.txt` is a text file containing two columns: one for temperature and one for the approximate time for explosion. The results from the run are summarized in the file `odtx-lx10.res`.

Examples

Two examples are now provided that show how the aforementioned codes are used. The first example shows how to calibrate for a series of ODTX data that fall in a straight line, while the second example shows how to use the codes in detail to perform an EZM calibration.

Example 1: Two-Point Calibration

In this example calibration, the experimental ODTX data for material ExpOne are shown in Figure 12 below. Since the data appear in a straight line, a standard Excel fit-curve relation to the data provides two calibration points at the two extremes of the temperature range: (300°C, 23.4 s) and (220°C, 5739 s). Assume that the thermal properties are the same as the default values in the input deck.

Temperature, °C	Time, sec
300	20
290	55
280	78
270	105
260	342
250	811
240	1254
230	2600
220	5000

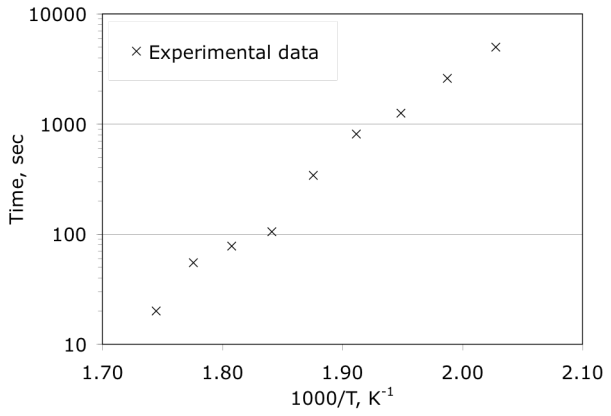

FIGURE 12. Example 1 explosive ODTX data.

Since the thermal properties are the same as default, then the input deck to `PT_inputmake.pl` is very simple:

```
matname ExpOne
odtx_1 300 23.4
odtx_2 220 5739
```

Note that this example is applied on the GPS cluster (and hence no `srun` command is needed) For a Linux cluster, the parameter `use_srun` should be set to 1. The input decks for ALE3D and the EZM calibrator is generated using the command

```
perl PT_inputmake.pl -i input_ex1.txt
```

where `input_ex1.txt` is a text file containing the above 3 lines. The EZM calibrator may then be run by typing

```
perl ezm_calibrator.pl
```

While the calibrator is running, the code will list a lot of error messages such as "No such file or directory" or "[0] MPI Abort by user Aborting program!" These messages are to be expected and the user should not be alarmed. After a while, the calibrator converges on the first point ($zrx \sim 12.6$), and then moves on to the second point, where it converges at ($zrx = 23.0$, $erx = 41844$). It will then re-run at the first point to check for the effect of the new *zrx* and *erx* on the first point (the convergence is reduced from 0.01 to 0.09). At the end of the run, the EZM calibrator output states the converged parameters (the aforementioned *erx* and *zrx*, and $m = 1$). These values have been automatically updated in the kinetics file (`lx10_kin_bd_v01.in`).

After the run, the calibrator has produced a large number of files that can be removed. This is accomplished by using the cleaner by typing

```
source cleaner
```

This removes the unwanted ALE3D files. If interested, the user may look at the various `*.out` files that describe the calibrator's attempt to converge for each of the three points.

A check on the calibrated kinetics curve can be produced using

```
perl autovary_odtx.pl -list data_ex1.txt
```

where `data_ex1.txt` is a text file containing the data table similar to that in Figure 12. This creates simulated ODTX times for each of the experimental temperature values. The resultant plotted simulated data are shown in Figure 13.

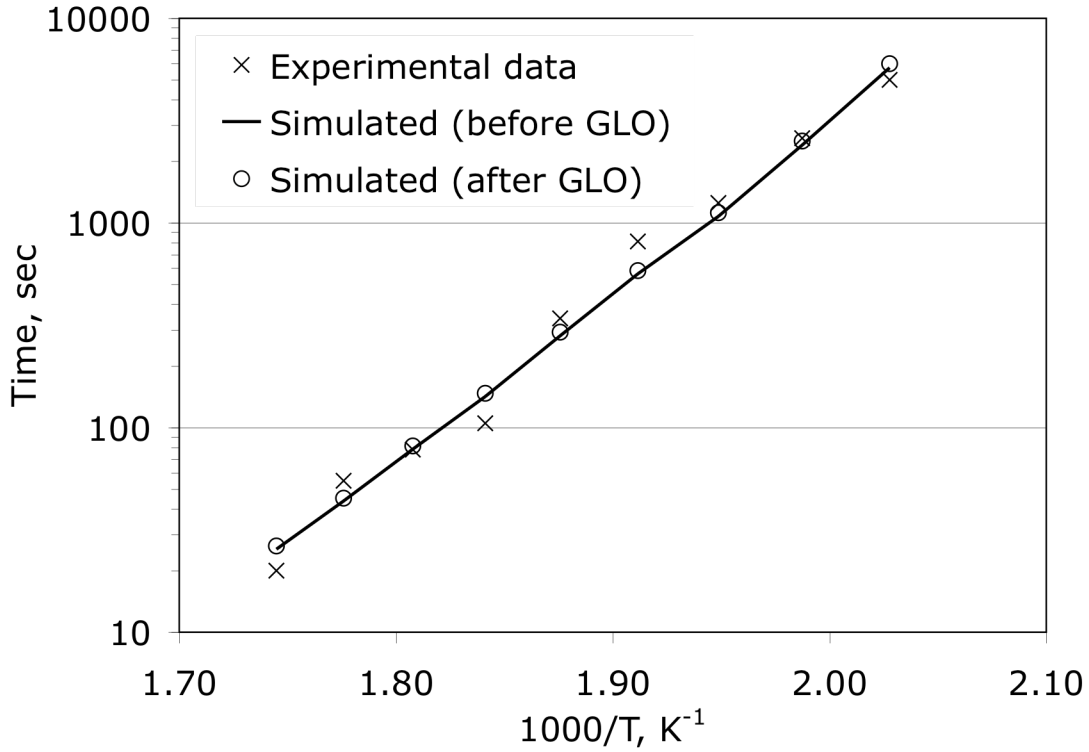

FIGURE 13. Simulated ODTX data using the EZM calibrator compared to experimental data for the example material.

To fine-tune the kinetic parameters using GLO, type at the command prompt

```
perl glomake.pl -datf data_ex1.txt -o ex1.gcf
mkdir GLO
cp ex1.gcf GLO
cp sphh.sami GLO
cp auto_cleanup.pl GLO
cd GLO
glo ex1.gcf
```

Once GLO has finished its run, the excess files in each directory may be removed by typing

```
perl auto_cleanup.pl
```

The GLO run has produced several directories beginning with the letter p (e.g. `p0001`). To determine which directory to keep, open the `ex1.u1` file and choose the iteration with the lowest figure of merit. The directory contains the values of the kinetic parameters in the `varvals` file and simulated ODTX times in the `results.res` file. Figure 12 shows that GLO does not significantly change the kinetic parameters and corresponding ODTX results for this example.

Example 2: Three-Point Calibration

This second example shows how to calibrate parameters when a bend exists in the data. The data, shown below in Figure 14, show a pronounced dogleg at 1000/T = 1.95. For this example, three ODTX points are used, and the value of *p* used is 2 to account for the bend. Assume that the material properties for this explosive, ExpTwo, are the same as the default parameters.

Temperature, °C	Time, sec
300	20
290	35
280	50
270	60
260	80
250	150
240	300
230	1000
220	5000

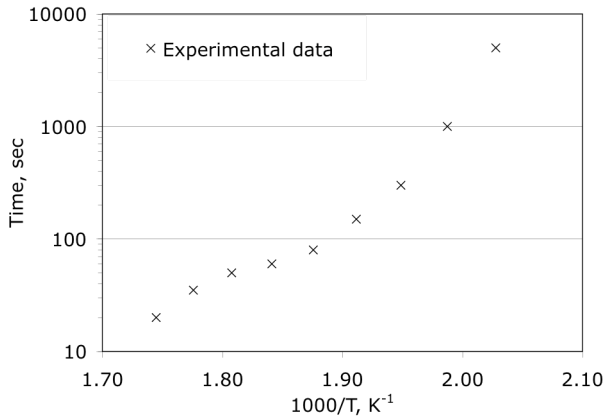

FIGURE 14. Example 2 explosive ODTX data.

First, three representative data points must be chosen. Here, we choose the two extremes of the temperature range and a point at the base of the bend. An example input for this explosive is provided below:

```
matname ExpTwo

pt_p 2
ignore_warn 1

odtx_pts 3
odtx_1 300 20
odtx_2 220 5000
odtx_3 250 150
```

Although running the EZM calibrator will converge to a solution on its own, here we will examine how to manually calibrate using `auto_optimizer.pl`. The execution of this perlscript for the first point is as follows:

```
perl auto_optimizer.pl -t 300.0 -g 20 -low 5 -high 30 -stepsize
1.e5
```

The script opens the kinetics file and determines the reactants and products for the default reaction `pt_solid_to_prod`. The external boundary temperature is set at 300°C, the maximum time step size is set to 0.01 sec, and the parameter *zrx* is varied between 5 and

30 until the sample explodes at 20 sec (+/- 1%). The resultant calibrated value of *zrx* for the first point is approximately 7.74.

The coupled *erx*-*zrx* calibration may also be implemented via this perlscript. If the above line were used to calibrate *zrx* for the first point (300.0 C, 20 sec), then the coupled *erx*-*zrx* algorithm may be implemented at a second point (220 C, 5000 sec), using the **-temp_pin** flag. Update the kinetics file with the value of *zrx* obtained for the first point, and then enter the following command line to calibrate *erx* and *zrx* for the second point using the coupling at the first point:

```
perl auto_optimizer.pl -t 220 -g 5000 -low 5 -high 30 -stepsize
1.e7 -temp_pin 300
```

This tells the script to attach the pre-calibrated values of *erx* and *zrx* to the temperature 300°C, thus establishing the coupling of the two parameters using Eq. (7), which is then implemented during each successive update to *zrx*. This calibrates *erx* and *zrx* to be approximately 42,121 and 18.4, respectively.

To complete the EZM calibration, update the kinetics file with the new *erx*-*zrx* pair, and then test the new parameters with the current value of *m* (*m* = 0 for this iteration) at the third point. Although there are several ways to achieve this, the easiest is by running `autovary_odtx.pl` at the third point temperature:

```
perl autovary_odtx.pl -init 250 -stepsize 1.e5 -time_limit 300
```

This results in an explosion time of 126 seconds for *m* = 0.

The user then adjusts the value of *m* in the kinetics file and repeats the above process until convergence is completed for the third point. This process becomes reasonably efficient with use of the history command at the unix prompt to re-run previous `auto_optimizer.pl` calls. A good general practice is to determine the bounds of *m*, update `ezm_calibrator.pl`, and then run the latter perl script. Repeating the above process with *m* = 1 yields the values *erx* = 45,821, *zrx* = 24.9 and a time to explosion of 206 seconds at 250°C. Therefore, `ezm_calibrator.pl` can be updated by changing the lines

```
$lowt = 126;
$hight = 206;
```

Running `ezm_calibrator.pl` leads to convergence with the following values:

- *erx* = 45697
- *zrx* = 22.6
- *m* = 0.3

Note the additional variables available using `ezm_calibrator.pl` compared to `auto_optimizer.pl`. Here, we provided known times for the limits of *m*. In addition, the user can give known calibrated values of *zrx* at the first point, and known calibrated values of *erx* and *zrx* at the second point. This provides a means to save time for restarting a calibration process. Because `ezm_calibrator.pl` calls `auto_optimizer.pl`, the latter's output files are created, in addition to the following:

- `opt-`*`i`*`-zrx.out`, where *`i`* is an integer: output from the auto_optimizer call for the first data point for iteration *i*.
- `opt-`*`i`*`-zrxerx.out`, where *`i`* is an integer: output from the auto_optimizer call for the second data point for iteration *i*.
- `opt-`*`i`*`-m.out`, where *`i`* is an integer: output from the auto_optimizer call for the third data point for iteration *i*.
- `resezm.res`: calculated times for the third data point for each iterated value of m
- `orig-`*`kinfile`*, where *`kinfile`* is the kinetics file name: a copy of the original kinetics file (modifications are made to the kinetics file during the calibration process)

Figure 15 below compares the simulated and experimental ODTX values for material ExpTwo. Further refinement of parameters may be done using GLO as described in Example 1.

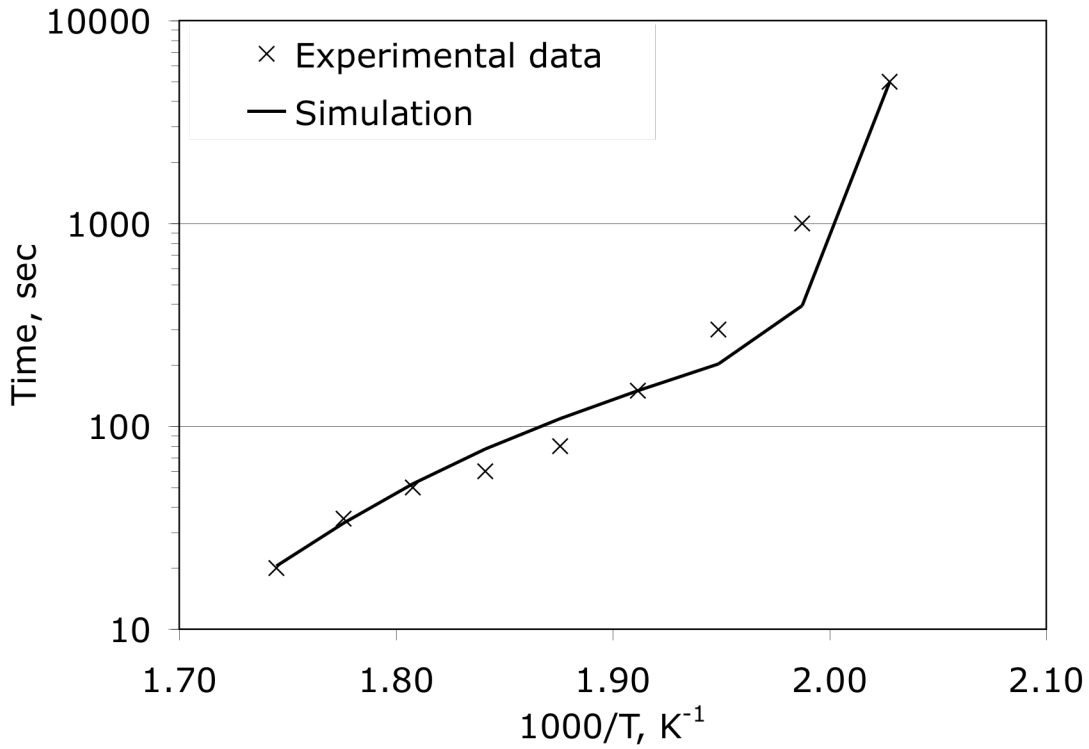

FIGURE 15. Simulated ODTX data using the EZM calibrator compared to experimental data for the material ExpTwo.

References

1. Wemhoff A. P. and Burnham A. K. (2006) Calibration Methods for ODTX Chemical Kinetics for Various Explosives, *Lawrence Livermore National Laboratory*, Report UCRL-TR-222032.

2. Burnham A. K. (2000) Application of the Sestak-Berggren Equation to Organic and Inorganic Materials of Practical Interest, *J. Therm. Anal. Cal*., Vol. 60, pp. 895-908.

3. Owens C., ed. (2005) LLNL Explosives Reference Guide, *Lawrence Livermore National Laboratory*, Report UCRL-WEB-217165.

4. Murphy M. J. (1999) GLO - Global Local Optimizer User's Manual, *Lawrence Livermore National Laboratory*, Report UCRL-MA-133858.

5. Wemhoff A. P. and Burnham A. K. (2006) Comparison of the LLNL ALE3D and AKTS Thermal Safety Computer Codes for Calculating Times to Explosion in ODTX and STEX Thermal Cookoff Experiments, *Lawrence Livermore National Laboratory*, Report UCRL-TR-220687.