We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter 6

Case Studies in Using MATLAB to Build Model Calibration Tools for Multiscale Modeling

Ricolindo L. Carino and Mark .F. Horstemeyer

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/62348

Abstract

This chapter illustrates the versatility of MATLAB for building interactive end-user software applications to support the pedagogy of a multiscale modeling approach to computational materials engineering. The case studies presented here demonstrate how preexisting codes that model complex material behavior, even if written in compiled computer languages such as Fortran or C++, may be utilized as computational libraries for model calibration software tools built with MATLAB. Intended for students in computational engineering (mechanics and materials), these tools execute on personal computers without MATLAB if the MATLAB Runtime shared libraries are installed. Publications coauthored by students using these tools to calibrate material models and to investigate the performance of engineering materials indicate that the tools enable advances in engineering design from a computational engineering perspective.

Keywords: multiscale modeling, interactive model calibration, parallel optimization, mixed-language programming, MATLAB

1. Introduction

The book "Integrated Computational Materials Engineering (ICME) for metals: using multiscale modeling to invigorate engineering design with science" by Mark F. Horstemeyer [1] aims to educate the next generation of practitioners of a simulation-based approach for the understanding, design, development, and manufacturing of load-bearing structural products. Intended as a textbook for senior undergraduate and graduate students of computational materials engineering, the book contains lecture notes, questions and solutions manual, and tutorials on the model codes at each length scale. The book has a companion website [2] with a

section each dedicated to the classes of materials (metals, ceramics, polymers, biomaterials, etc.) and to material models at the different length scales. Many of the model codes discussed in the book have been incorporated into compiled MATLAB [3] applications for students to run on personal computers and research workstations with or without MATLAB installed. The design of these MATLAB applications and examples of their use as computational tools to investigate materials for engineering products is the subject of this chapter.

Section 2 provides an overview of the multiscale modeling approach to the study material behavior. Section 3 describes the general design considerations and requirements for model calibration tools built on top of the preexisting model codes for multiscale modeling. The first MATLAB application built by the authors based on these requirements is DMGfit [4] for calibrating the internal state variable (ISV) damage and plasticity model [5, 6] written as a Fortran subroutine. Section 4 describes DMGfit in some detail as its support for user interactivity and its exploitation of multiprocessing capability in hardware heavily influenced the development of subsequent applications built by the authors. Section 5 describes TPgui [7], a graphical user interface (GUI) and calibration tool for a thermoplastic model of polymers [8]. TPgui was developed at about the same time as the underlying model; hence, flexibility was built into the interface to accommodate model revisions. Section 6 focuses on VPSCgui [9], a GUI to the viscoplastic self-consistent (VPSC) model of polycrystalline aggregates [10]. VPSC is a self-contained Fortran program, raising the issue of information exchange with the interface. Section 7 provides screenshots and summaries of the calibration tool for the multistage fatigue (MSF) model of crystal plasticity [11] and the modified embedded atom method (MEAM) parameter calibration (MPC) tool [12] that embeds LAMMPS-MEAM (large-scale atomic/molecular massively parallel simulator with the MEAM package for many-body potentials) [13–15]. Section 8 summarizes some lessons learned in building MATLAB applications for pedagogy and research in the multiscale modeling of materials.

2. Multiscale modeling of materials

Consider an engineering metallic product that is a component of a larger system. Of particular interest is a reliable prediction for the failure of the component. The simulation-based design of the component to satisfy specified engineering objectives would use information provided by material models at all involved length scales. See **Figure 1** for an illustration.

The larger system is the vehicle, with the automotive control arm as the component of interest. Pertinent questions regarding the component may include the following: As designed, where will failure occur, and what is the expected lifetime of the component? Can the component design be optimized and/or can the component be built using a different material such that the component will cost less, weigh less, and last longer? These questions may be investigated using a hierarchical multiscale model. Based on the illustration in **Figure 1**, the length scales and some of the computational models that have been used at each scale are listed in **Table 1**.

http://dx.doi.org/10.5772/62348

Figure 1. Multiscale modeling example of an automobile component made from metal (from [1], p. 11).

Length scale	Computational model
L1: Electrons (Å)	M1: Electronics principles (DFT)
L2: Atoms (nm)	M2: Atomistic (EAM, MEAM, MD, MS)
L3: Dislocations (100 nm)	M3: Dislocation dynamics
L4a: Grains (1 μm)	M4a: Crystal plasticity (1 μm ISV+FEA)
L4b: Grains (10–100 μm)	M4b: Crystal plasticity (10–100 μm ISV+FEA
L4c: Grains (100–500 μm)	M4c: Crystal plasticity (100–500 μm ISV+FEA)
L5: Macroscale continuum	M5: Macroscale ISV
L6: Component	M6: ISV
L7: Whole system	M7: ISV

FEA, finite element analysis; MD, molecular dynamics; MS, molecular statics.

Table 1. Multiscale modeling length scales and computational models.

The computational material codes typically solve complicated physics-based formulas for material behavior and are written in procedural computer languages, such as Fortran, C, or C++, or may even be mixed-language programs. These material model codes usually require an input file containing values of the model parameters and computational settings. Output files generated by these model codes are typically postprocessed or visualized by other software.

Computational simulations at lower-length scales generate information that will be used by material models at the higher-length scales (upscaling). Alternatively, models at higher-length scales may specify information that imposes boundary conditions on simulations at the lower-length scales (downscaling). The upscaling and downscaling calculations form "bridges" between material models and simulations. **Table 2** lists the model-bridging calculations based on the illustration in **Figure 1**.

Model #1	Model #2	Bridging calculation	Model #1	Model #2	Bridging calculation
M1	M2	Bridge 1=Interfacial energy; elasticity	M1	M5	Bridge 6=Elastic moduli
M2	M3	Bridge 2=Mobility	M2	M5	Bridge 7=High rate mechanisms
M3	M4a	Bridge 3=Hardening rules	M3	M5	Bridge 8=Dislocation motion
M4a	M4b	Bridge 4=Particle interaction	M4a	M5	Bridge 9=Void/crack nucleation
M4b	M4c	Bridge 5=Particle-void interactions	M4b	M5	Bridge 10=Void/crack growth
M4c	M5	Bridge 11=Void-crack interactions	M5	M6	Bridge 12=FEA
			M6	M7	Bridge 13=FEA

Table 2. Model-bridging calculations in multiscale modeling.

3. Design considerations for model calibration tools

The applications described here are intended to support pedagogy and research in a multiscale modeling approach to computational materials engineering. Because students who are learning complex models of material behavior will use the applications, interactivity is an important consideration. In addition, automation is also important so that the applications can save time for busy researchers who are already familiar with the models and just need to calculate a few sets of model parameters. Students may be using personal computers running Microsoft Windows or Mac OS or research workstations running Linux. Whatever is the operating system, most modern laptops and workstations already have built-in multiprocessing capabilities that may be harnessed to speed up intensive computations. In addition, the model codes may undergo revisions as research progresses. Finally, the application may need to be shared with collaborators for exchange of ideas. Thus, many issues pertaining to students, research collaborators, computing platforms, and the certainty of incremental model revisions need to be addressed in the design of applications for both pedagogy and research in multiscale modeling.

http://dx.doi.org/10.5772/62348

At the core of the multiscale modeling approach are mathematical constitutive models representing material behavior at various length scales. These models are typically expressed as complex mathematical functions with several parameters. For a specific material, some of these model parameters may be known from the scientific literature or from the material data sheet prepared by the manufacturer, whereas other parameters are to be determined from material characterization experiments or from results of other computational simulations. The process of determining model parameters from experimental or simulation data is referred to here as model calibration.

Figure 2 depicts an interactive model calibration process. The model parameters produced as outputs of the process may be used in the model-bridging calculations in **Table 2** or by models in the higher-length scales. The model calibration tools described here were initially developed for students of computational engineering. A student may use a tool to complete an assignment, to model experimental data that are collected during a research project, or to learn about and revise its underlying material model.

Interactivity in a model calibration tool is a requirement so that end-users are able to immediately visualize the effect of changing a model parameter. This is important, as an incremental and heuristic strategy to calibration is sometimes necessary, owing to the complexity of a

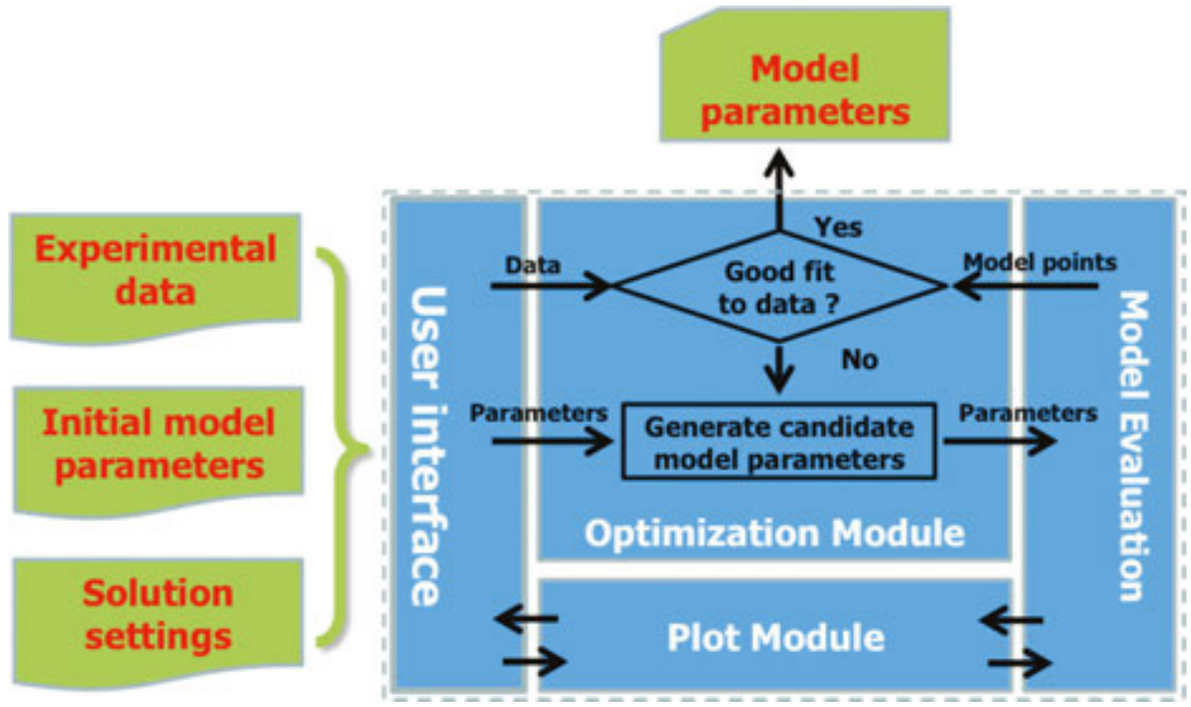

Figure 2. Interactive model calibration to determine model parameters from experimental data. The user specifies the experimental data, initial model parameters, and solution settings (the inputs) through the User interface. The Plot Module invokes the Model Evaluation module (the model code) to calculate a model curve from the model parameters currently on display. The model points are plotted alongside the experimental data. Individual parameters may be manually edited on the User interface to change the shape of model curve. The Optimization Module attempts to automatically find values for a set of parameters (the output) that will generate model curves that are "close" to the experimental data.

material model. Another requirement is that a calibration tool must run on stand-alone personal computers commonly used by students (usually Microsoft Windows or Mac OS) and on research workstations (typically Linux). Further, because modern personal computers and workstations typically have built-in multiprocessing capabilities, another design requirement is that the tool must be able to exploit parallelism available in the runtime system for the faster execution of applicable steps in the interactive model calibration process.

The scientific theory underlying a material model is always a work in progress. Revisions to theory translate into changes in the constitutive model code, such as the addition of model parameters or extended formulas to better capture material behavior. Therefore, another design requirement for the tool is that code changes to accommodate additional model parameters or updates to model formulas should be confined to the Model Evaluation module. This will enable students who are doing research on improving the underlying material model to concentrate on the model code with very minimal changes in the Model Evaluation module and the User Interface.

Research collaborations typically involve the transmission of software for evaluation or trial use by interested parties. In many cases, constitutive model codes written as part of sponsored research have disclosure restrictions. To facilitate the exchange of novel ideas embedded in material models without disclosing source codes, the binary executable for a model calibration tool will have to be transmitted. Thus, the programming environment for the tool must be capable of building a binary executable that is royalty free when redistributed.

MATLAB was selected as the programming environment for building the model calibration tools because of the following features:

1. Availability of numerical libraries, graphical functions, parallel code development, and GUI-building tools in a single environment;
2. Constitutive model codes written in other languages (Fortran, C/C++) can be integrated without rewrite with MATLAB code;
3. The same MATLAB application source code runs on multiple operating systems (Windows, Linux, and Mac OS); and
4. The MATLAB application can be compiled into an executable that is freely redistributable.

The succeeding sections describe some of the model calibration tools built using MATLAB by the authors primarily for graduate research assistants in the Center for Advanced Vehicular Systems (CAVS), Mississippi State University. These tools are interactive and at the same time support a semiautomated process of model calibration. The tools run on personal laptops and desktop workstations with or without MATLAB and can exploit multiprocessing capabilities provided by hardware. For some model codes that are subroutines written in Fortran only or in C/C++ with Fortran, MATLAB gateway functions (mexfunctions) were written to facilitate the invocation of the subroutines as library calls from the Plot and Optimization modules of the tool. For other single-program model codes that read input decks and write output to text files, MATLAB functions were written to generate the input decks from entries in the User Interface and to extract relevant values from the program outputs. MATLAB-generated binary

http://dx.doi.org/10.5772/62348

executables of the tools have been used by collaborators and other interested researchers. Each tool also has a website that provides usage instructions and links to download the executable or the sources if made publicly available.

4. DMGfit—damage and plasticity model-fitting tool

DMGfit is an interactive calibration tool for the ISV damage and plasticity model [5, 6]. The model is written in Fortran as an ABAQUS User Material subroutine (UMAT or VUMAT) [16]. DMGfit executes the UMAT/VUMAT as a library routine, not as a separate external process. DMGfit inputs comprise experimental stress-strain data to determine "material properties" that are subsequently used along with the UMAT/VUMAT by an ABAQUS simulation in length scale L6. **Figure 3** depicts the component modules of DMGfit, its interface to the model code UMAT/VUMAT, and how a finite element simulation consumes the DMGfit output. The source codes of DMGfit are online [17].

The damage and plasticity model is specified by 55 parameters (material properties) at the time of this writing. Some properties, such as bulk modulus, shear modulus, and melting temperature, are fixed constants for a given material and may be obtained from the literature. Other properties, such as average radius of voids, average size of particles, and average grain

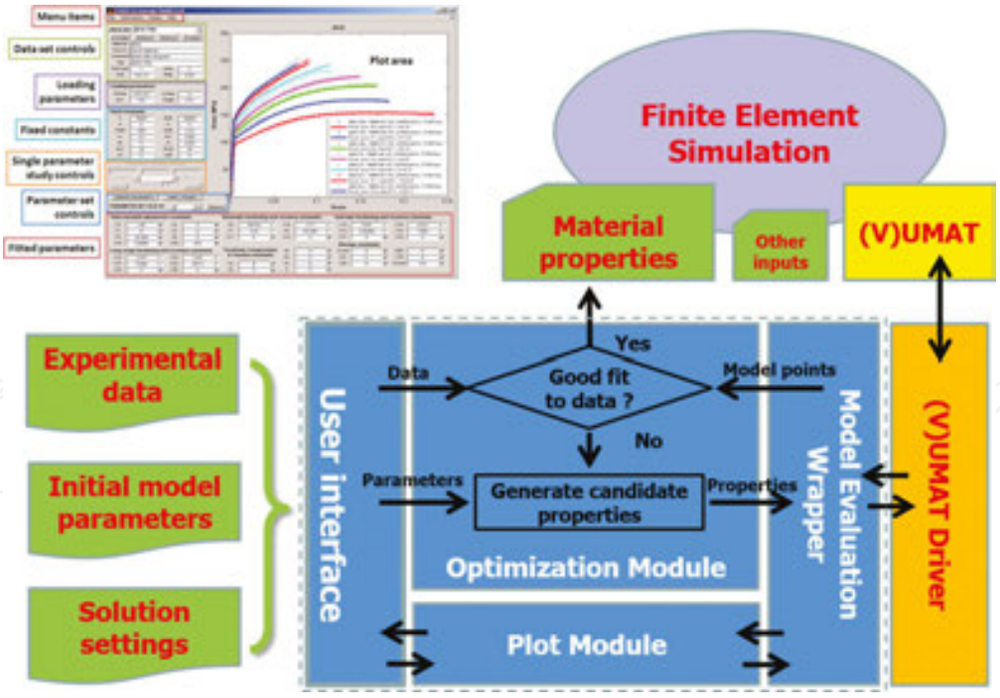

Figure 3. DMGfit screenshot, component modules (blue boxes, in MATLAB), damage and plasticity model subroutine (yellow, in Fortran), and model driver (orange, in Fortran). Experimental data inputs to DMGfit are stress-strain curves. The Material properties output by DMGfit and the model subroutine are inputs to a finite element simulation in ABAQUS. The bridge between MATLAB and Fortran is a mexfunction in the Model Evaluation Wrapper.

size, are measured from characterization experiments on samples of the material. DMGfit calibrates the remaining properties using stress-strain data collected from tension, compression, and shear experiments on the samples at various combinations of strain rates and temperatures.

The interactive calibration of the damage and plasticity model using DMGfit typically proceeds as follows:

1. Load all experimental data sets. For each data set, establish the experiment settings (initial temperature, strain rate, stress units, etc.), loading parameters, and fixed constants.
2. Start by fitting the experimental data set with the lowest temperature and lowest rate. Temporarily exclude the rest of the data sets. If there are different tests, fit the compression data sets first followed by tension data sets and then torsion data sets.
3. For the first data set, adjust the parameters as follows: yield C3; kinematic hardening C9 and dynamic recovery C7; isotropic hardening C15 and dynamic recovery C13.
4. Restore second data set. If it has a different temperature than the first, adjust the parameters as follows: {C3, C4} if yield is temperature dependent, then {C10, C8, C16, C14}. If the data set has a different strain rate, adjust C1 and C5 if yield is strain rate dependent, then {C9, C7, C11} and {C15, C13, C17}.
5. Repeat step 4 for the rest of the data sets. If adjusting the temperature dependence parameters (even Cs) does not produce good models for high temperature data, adjust C19 and C20. Adjust torsion, compression, and tension differentiation parameters, if adding stress state-dependent experimental data.
6. Adjust damage parameters. Readjust parameters in other boxes as necessary.
7. Create a "restart" file to record calibration session for resumption later. Merge the material constants with an existing ABAQUS input deck if one has been prepared previously or write results to text files for postprocessing by other applications such as Microsoft Excel.

The user may specify the number of plots to be displayed on the interface. There are three plotting strategies: a single plot for all data sets, which may result in an overcrowded plot area; one data set per plot, which may produce many small plots; and, as a compromise, one plot per data set type (i.e., one plot for tension data sets, another plot for compression data sets, etc.). After a data set is loaded into the application, it may be included or excluded from plots and from participating in the calibration process. See [4] for details about all features of DMGfit. **Figure 4** depicts a sample screenshot of DMGfit when used to calibrate the material properties for a 7075-T651 aluminum plate [18].

DMGfit provides three methods of adjusting the model parameters. First, a user can manually adjust model parameters by directly editing their values and clicking the "Apply changes" button to regenerate the model curves. Second, a user can activate the "Parameter study" slider by a right click on a parameter. A click or drag on the slider will vary the parameter and regenerate the model curves. The third method allows two or more parameters to be adjusted

http://dx.doi.org/10.5772/62348

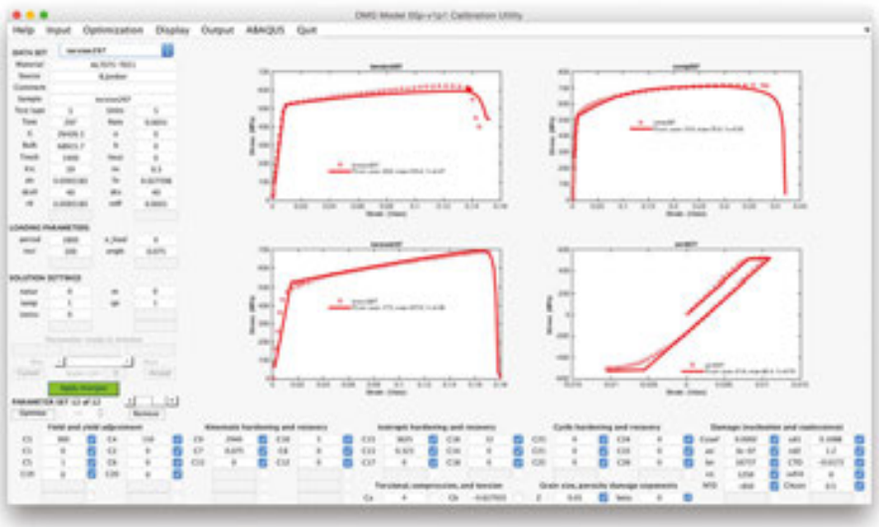

Figure 4. DMGfit screenshot showing the material properties for an aluminum 7075-T651 plate [18]. The damage and plasticity model parameters calibrated by DMGfit predict the strength, failure, and other mechanical characteristics of the plate.

simultaneously using optimization. The optimization objective is to minimize the distance between the model curve and the experimental stress-strain data. The optimization variables are the unchecked parameters. Clicking the "Optimize" button runs the optimization using the displayed values of the unchecked parameters as the starting solution. The optimization methods available in DMGfit are the MATLAB functions **fminsearch** (simplex search), **lsqnonlin** (nonlinear least squares) **fmincon** (constrained minimization), **patternsearch** (pattern search optimization), and **ga** (genetic algorithm).

DMGfit uses multiple computational cores depending on the settings of the menu item "Optimization | Enable parallelism" and pop-up menu beside the Optimize button. This pop-up determines the number of starting solutions when the Optimize button is clicked. If one starting solution is specified (i.e., "Optimize 1") and the optimization method is **fmincon** or **patternsearch** or **ga**, then DMGfit will use the multiple cores. These optimization methods have parallel implementations in MATLAB; hence, the optimization process will benefit from the multiple cores. The methods **fminsearch** and **lsqnonlin** do not have parallel implementations (in MATLAB R2012a), so optimization with these methods will not benefit from multiple cores with "Optimize 1"; however, **fminsearch** and **lsqnonlin** will still find final solutions. If "Optimize N" is specified for a small value of N, then DMGfit will automatically generate a number of starting solutions as follows. Let K be the number of parameters that are unchecked in the DMGfit user interface. The "global" search space for the optimization is the K-dimensional hyperrectangular region R:

$$R = [\min\_1, \max\_1] * [\min\_2, \max\_2] * \ldots * [\min\_K, \max\_K],$$

where [min_i, max_i] is the range for the *i*th optimization variable. DMGfit will divide each range into *N* subintervals so that there will be *N*^*K* (*N* to the power *K*) hyperrectangular subregions. DMGfit will generate a random starting solution within each subregion and run an optimization from the random starting point. The optimization variables will be bounded by the limits of the subregion (except if the method is **fminsearch**). The "spmd…end" (single program, multiple data) MATLAB statement will be used to execute the *N*^*K* optimization runs in parallel. Because one run might take several seconds or a few minutes, the parallel run may still require a significant wait. DMGfit will collect a number of "locally good" results along with the limits of the subregions returning such results. When all *N*^*K* searches are complete, the results can be browsed manually to decide which are "really good". Not all subregions will produce "good" results; hence, much less than *N*^*K* results will be returned.

5. TPgui—a flexible GUI and fitting tool for a thermoplastic polymer model

Many engineering products include components made from polymers. A modern automotive vehicle, for example, has polymer parts such as plastics, rubbers, fibers, foams, and adhesives. Thus, it is important to predict the mechanical responses of polymer components, as these may be subjected to high strain rates during crash scenarios. In general, there are three groups of polymers: thermosets, which are rigid materials that do not flow under the action of heat; thermoplastics, which become fluid when heat is applied; and, elastomers, which can be easily deformed but will return to the original size when the loading is released. This section briefly describes TPgui, a GUI and calibration tool for an ISV model for thermoplastics. **Figure 5** provides a TPgui screenshot. The model equations are described in [8], and a TPgui tutorial may be downloaded from [19].

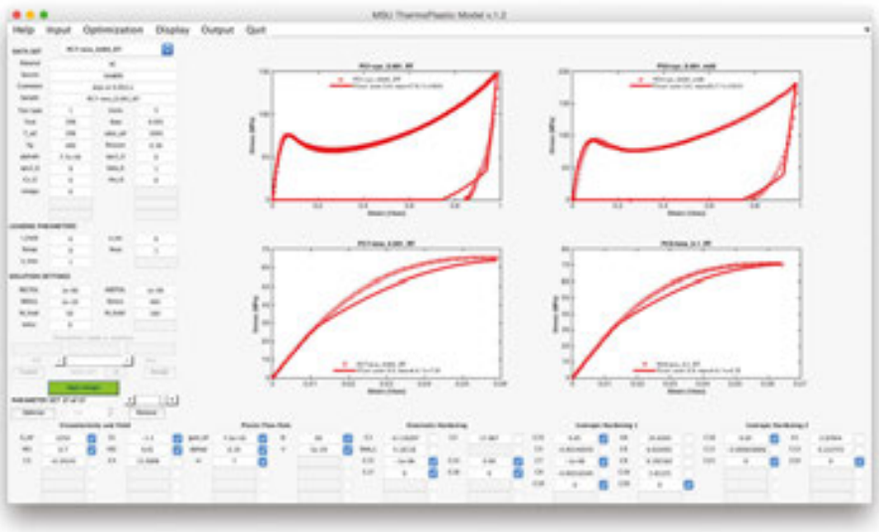

Figure 5. TPgui screenshot showing the material properties for a polycarbonate.

http://dx.doi.org/10.5772/62348

The TPgui user interface code is derived from DMGfit; hence, TPgui inherits all the interactive and parallel-enabled features of DMGfit. Unlike the model code of DMGfit that is a Fortran subroutine, the ISV thermoplastic model code is written as a MATLAB function; therefore, a mexfunction is not necessary in TPgui. A theoretician was developing the thermoplastic model code at the same time the authors were building the user interface. Therefore, to accommodate additional model parameters, experimental data, and computational settings, the TPgui interface was designed with placeholders for items that were yet to be specified by the theoretician. These placeholders are the gray (disabled) textboxes in **Figure 5** under the "DATA SET" label for experimental data, under the "SOLUTION SETTINGS" label for computational settings, and on the lower portion of the user interface for model parameters. The interface allows up to 60 model parameters arranged in a 6×10 grid. More placeholders for model parameter may be added if necessary by copy-pasting the bottom row in MATLAB's GUIDE (GUI Design Environment). To activate, for example, the placeholder in grid positions (4,1) and (4,2) (below C3 and C4, respectively) for new model parameters "C3a" and "C4a", the initialization routine only have to be edited like the highlighted lines in following code fragment:

```
% Property fields (for fitted model parameters)
% Format: 'handle', 'gui_string', 'prop. name', 'init fixed? min max',
'description/tooltip', 'TeX annotation'
% An empty handle or gui_string means it will not appear in the GUI
property_strings = { ...
    % columns 1 & 2    (Viscoelasticity and Yield)
    'p1',  'E_ref',  'E_ref', '0 0 0 1.0e6', 'Reference elastic
modulus (MPa)', 'E_{ref}'; ...
    'p2',  'E1',     'E1',    '0 0 0 10.0', 'Activation term for
elastic modulus temperature dependence (MPa)', 'E1'; ...
    'p11',  'VE1',    'VE1',  '0 0 0 5.0', 'Viscoelastic parameter
(numerator) (MPa)', 'VE1'; ...
    'p12',  'VE2',    'VE2',  '0 0 0 1.0', 'Viscoelastic parameter
(denomenator) (MPa)', 'VE2'; ...
    'p21',  'C3',   'Yref',   '0 0 -0.9 0',  'Activation term for
yield temperature dependence (MPa)', 'C_3'; ...
    'p22',  'C4',     'Y1',   '0 0 0 100',  'Temperature independent
yield (MPa)', 'C_4'; ...
    'p31',  'C3a',      ' C3a ', '0 0 1 0', 'New parameter C3a', '
C_{3a}'; ...
    'p32',  '',     '', '', '', ''; ...
    'p41',  'C4a',      ' C4a ', '0 0 1 0', 'New parameter C4a', '
C_{4a}'; ...
    'p42',  '',     '', '', '', ''; ...
    % ''
    'p51',  '', '', '', '', ''; ...
    'p52',  '', '', '', '', ''; ...
    % columns 3 & 4    (Plastic Flow Rule)
```

Similar edits may be used to activate placeholders for experimental data and computational settings. Column labels for related model parameters are also customizable. To use the new parameters "C3a" and "C4a" in the thermoplastic model, the initializations in model code only have to be edited like in the following fragment:

```
function [time strain stress statev] = TP_Model_Uniaxial(dataset,
property, option)
% ISV model for Thermoplastics.

%---------- Y
C3 = property.Yref(1);      % Yref  [...]
C4 = property.Y1(1);        % Y1  [...]
%---------- New parameters C3a, C4a
C3a = property.C3a(1);      % C3a [...]
C4a = property.C4a(1);      % C4a [...]
```

TPgui demonstrates the versatility of MATLAB in facilitating the development of dual-purpose model calibration tools. The TPgui code is a result of a strategy for building a flexible user interface that supports interactive calibration of model parameters by student researchers and at the same time serves as a test environment for theoreticians investigating alternate formulations of the underlying model. Just like an end-user being able to immediately visualize the effect of changing the value of a model parameter on the model curve, a theoretician can revise a model formula and immediately visualize its effect on the model curve.

6. VPSCgui—VPSC model calibration interface

VPSC is a Fortran 77 program that simulates the plastic deformation of polycrystalline aggregates subjected to external strains and stresses. VPSC stands for viscoplastic self-consistent and refers to the particular mechanical regime addressed (VP) and to the approach used (SC). VPSC accounts for the following material behavior: full anisotropy in properties and response of the single crystals; the hardening, reorientation, and shape change of individual grains; and grain interaction effects. In addition to providing the macroscopic stress-strain response, VPSC predicts the evolution of hardening and texture associated with plastic forming. The simulation procedure can be applied to deformation of metals, intermetallics, and geologic aggregates [10].

As of version 7b, the VPSC code is text-based and is executed from a command line. One or more of four input files have to be manually edited before a computational deformation experiment can be started. The program produces up to 10 output files; the files containing results of interest have to be postprocessed and transformed as inputs for other applications for visualization. In addition, manual calibration of the model parameters to produce a stress-strain curve that will match an experimental stress-strain data set is a very tedious, mechanical, and error-prone undertaking. Such was the experience of graduate students when they first used the VPSC to model deformation of magnesium and thus motivated the development of a user-friendly GUI to VPSC that incorporates a model-fitting functionality.

VPSCgui, a GUI to the VPSC executable, was built by the authors with the following requirements: it must be "point and click"; it must incorporate an interactive model calibration functionality; and revisions to the VPSC source code must be minimal, with no changes to the program logic. The source code, sample input, and documentation for VPSCgui are online [9], excluding the VPSC sources. The only changes to the VPSC code to make it work with the interface involved the renumbering some of the I/O units, and the addition of write statements

http://dx.doi.org/10.5772/62348

at several places in the code so that VPSC will write "STOPPED" to standard output just before it ends as a completion signal to the interface. **Figure 6** is a screenshot of VPSCgui in modeling the behavior of magnesium AM30 undergoing channel die compression [20].

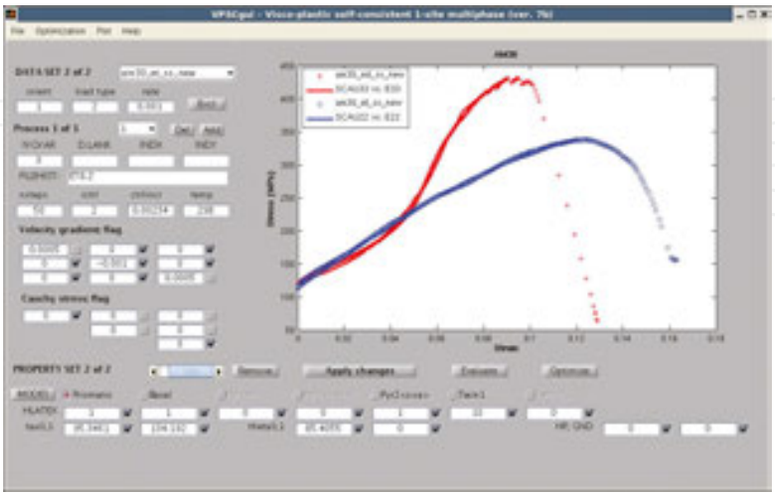

Figure 6. VPSCgui screenshot: experimental data controls (top left), deformation simulation settings (middle left), model parameters (bottom), experimental stress-strain data plots (discrete points), and VPSC model plots (solid lines).

VPSCgui implements the interactive model calibration process outlined by **Figure 2**. Unlike DMGfit and TPgui that execute their underlying models via library calls, VPSCgui invokes the VPSC program as a separate external process, and the two processes communicate through the relevant input and output files of VPSC. The input files are initially loaded into VPSCgui where a user can edit the model parameters and specify the settings for a deformation simulation. When the user clicks the "Evaluate" button, the input files are updated with the changes made on the interface, and the VPSC executable is invoked to run the deformation simulation. When the VPSC simulation completes, the interface retrieves and displays the model curves from the output files. The "Optimize" button follows the same sequence, where the optimization algorithm automatically adjusts user-selected model parameters.

VPSCgui demonstrates how MATLAB can be used to build a model calibration interface for preexisting model code (VPSC) that is a self-contained program. In this case, a mexfunction is not applicable, as VPSC has to execute as a process that is separate from VPSCgui. Data must be exchanged through the input and output files of VPSC; therefore, VPSCgui includes routines to read/write the input files of VPSC and to read the relevant output files for the model curves. Developing these routines for VPSCgui required much effort, as the Fortran subroutines in VPSC to read the input files were practically translated into MATLAB.

VPSCgui also exploits multiprocessing capability provided by hardware. Because one deformation simulation is required to model a single data set, parallelism occurs when there are several data sets being modeled. The VPSC program is inherently serial; however, several VPSC instances can run in parallel, one instance per data set. VPSCgui invokes the VPSC instances in separate directories to avoid collisions when writing the output files, as the file

unit numbers will be the same across instances. After parallel invocation, VPSCgui periodically checks the files that are piped from the standard output of each VPSC instance for the "STOPPED" signal.

7. Other model calibration tools

Experience gained in building the model calibration tools described in the previous sections guided the development of more tools for other length scales. This section briefly describes the calibration tools for the MSF model and the MEAM.

Figure 7 shows a screenshot of the MSF model calibration tool [11]. The MSF model predicts the amount of fatigue cycling required for the appearance of a measurable crack, the crack size as a function of and loading cycles. The model incorporates microstructural features that affect the fatigue life predictions for incubation, microstructurally small crack growth, and long crack growth stages in both high-cycle and low-cycle regimes [21].

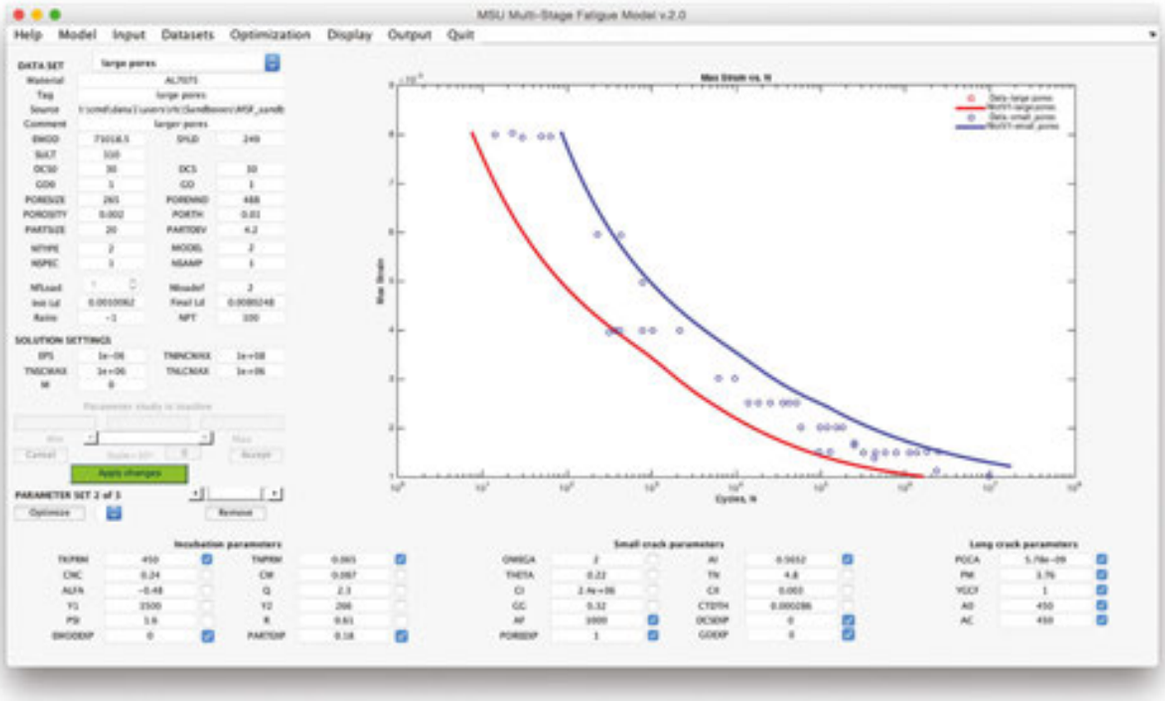

Figure 7. Screenshot of the MSF model calibration tool with model parameters to predict the fatigue life for aluminum 7075-T651 [18].

The MSF model code is an ABAQUS VUMAT written in Fortran, similar to the damage and plasticity model used by DMGfit. The MSF interface implements the same interactive and parallel features of DMGfit.

http://dx.doi.org/10.5772/62348

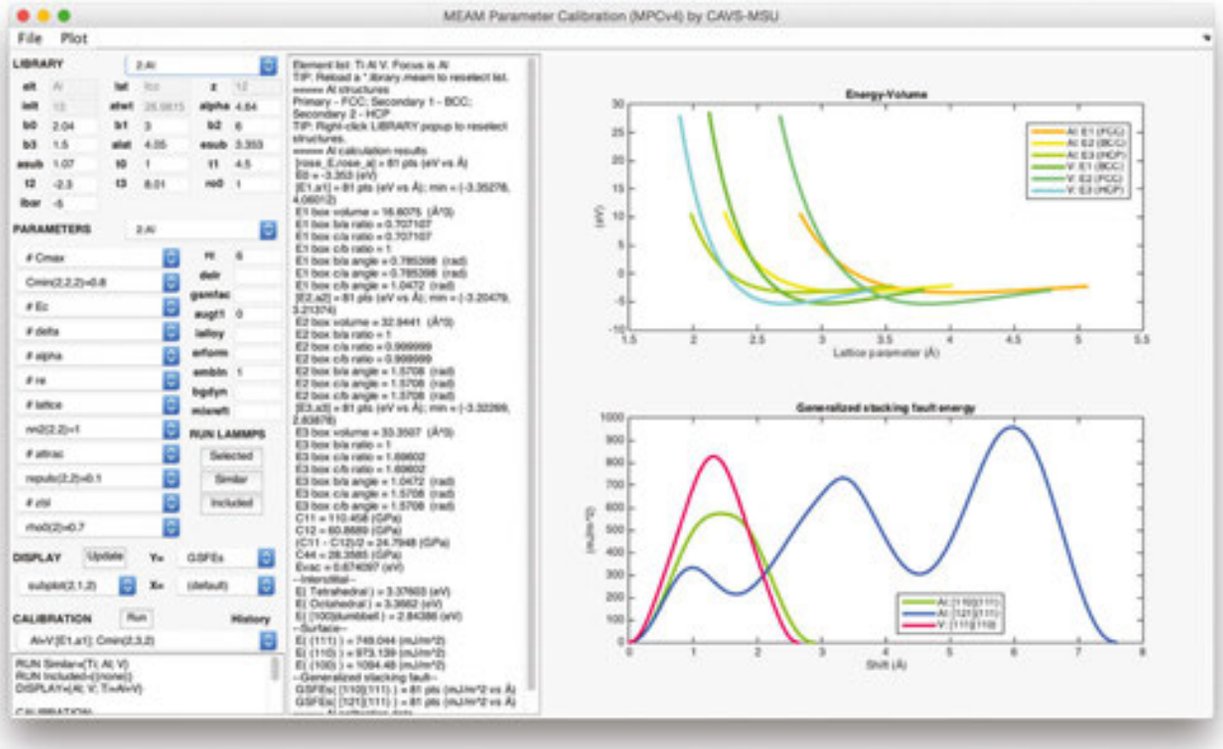

Figure 8. Screenshot of MPC tool.

Figure 8 shows a screenshot of MPC tool [12] for the interactive editing of MEAM library and parameter files and for the semiautomated calibration of MEAM parameters. The calibration targets may be density functional theory (DFT) simulation data and/or experimental data. Similar to VPSCgui, MPC reads and writes the LAMMPS input files; however, unlike VPSCgui, MPC does not execute the model code as a separate external process. Instead, MPC executes LAMMPS-MEAM, the large-scale atomic/molecular massively parallel simulator (in C++/C) with the MEAM package for many-body potentials (in Fortran) [15] as a library call such as in DMGfit.

A prior version of MPC [22,23] invokes a Python script that in turn executes LAMMPS-MEAM as a separate external program. The Python script also retrieves the relevant information to be returned to MPC from the LAMMPS-generated log file. This strategy of using Python as an intermediary between MPC and LAMMPS incurs significant file I/O overhead. Further, it was cumbersome to revise the Python scripts to set up additional LAMMPS calculations. As a consequence, the current MPC version is designed to use mexfunctions for invoking LAMMPS-MEAM as a library call, eliminating the need to run the Python interpreter and significantly reducing file I/O overhead.

8. Concluding remarks

Several issues need to be addressed when building a model calibration tool for a preexisting material model code. A potential end-user of the tool may be a student learning about the model, a researcher who needs the tool to model experimental data for some new material, or a theoretician seeking to improve the underlying model formulas so that it better captures material behavior. In each case, interactivity is a very important feature of the tool. A user may have a choice between a Microsoft Windows and a Mac OS personal machine to run the tool, or only a Linux workstation may be available. The model code may be written as a Fortran subroutine, a MATLAB script, or as a complete stand-alone mixed-language program.

The following MATLAB features were found to be sufficient in addressing all of the aforementioned issues. MATLAB GUIDE enables the creation of interfaces that support an interactive and semiautomated model calibration process. MATLAB Optimization Toolbox provides a variety of optimization techniques for automatically adjusting selected model parameters to fit experimental data. MATLAB Parallel Computing Toolbox enables the writing of parallel code that exploits multiprocessing features of modern personal computers to accelerate the model calibration process. MATLAB MEX files enable model codes written as Fortran subroutines or C/C++ functions to be invoked directly by MATLAB. In addition, model codes that are stand-alone programs can be executed as external processes through the MATLAB **system()** command. The model calibration tools described in this chapter demonstrate the versatility of MATLAB as a programming environment for building such tools.

Author details

Ricolindo L. Carino[1*] and Mark .F. Horstemeyer[1,2]

*Address all correspondence to: carino@cavs.msstate.edu

1 Center for Advanced Vehicular Systems, Mississippi State University, Mississippi, USA

2 Department of Mechanical Engineering, Mississippi State University, Mississippi, USA

References

[1] Horstemeyer MF. Integrated Computational Materials Engineering (ICME) for Metals: Using Multiscale Modeling to Invigorate Engineering Design with Science. Hoboken: John Wiley & Sons; 2012. 472 p. DOI: 10.1002/9781118342664p.

[2] Mississippi State University Center for Advanced Vehicular Systems. Engineering Virtual Organization for CyberDesign [Internet]. 2011 [Updated 2014]. Available at:

https://icme.hpc.msstate.edu/mediawiki/index.php/Main_Page [Accessed: 2016-01-14].

[3] MathWorks. MathWorks—MATLAB and Simulink for Technical Computing [Internet]. 1994 [Updated 2016]. Available at: http://www.mathworks.com [Accessed: 2016-01-14].

[4] Cariño RL. DMGfit User Guide [Internet]. 2012 [Updated 2015]. Available at: https://icme.hpc.msstate.edu/mediawiki/index.php/DMGfit_55p_v1p1 [Accessed: 2016-01-14].

[5] Bammann DJ, Chiesa ML, Horstemeyer MF, Weingarten LI. Failure in ductile materials using finite element methods. In: Jones N, Wierzbicki T, editors. Structural Crashworthiness and Failure. Barking (England): Elsevier Science Publishers Ltd.; 1993. pp. 1–54.

[6] Horstemeyer MF, Lathrop J, Gokhale AM, Dighe M. Modeling stress state dependent damage evolution in a cast Al-Si-Mg aluminum alloy. Theoretical and Applied Fracture Mechanics. 2000;33(1):31–47. DOI: 10.1016/S0167-8442(99)00049-X.

[7] Bouvard JL, Cariño RL. CMD Codes Repository—TPgui [Internet]. 2010 [Updated 2012]. Available at: https://icme.hpc.msstate.edu/viewvc/CMD%20Codes%20Repository/TPgui/ [Accessed: 2016-01-14].

[8] Bouvard JL, Ward DK, Hossain D, Marin EB, Bammann DJ, Horstemeyer MF. A general inelastic internal state variable model for amorphous glassy polymers. Acta Mechanica. 2010;213(1):71–96. DOI: 10.1007/s00707-010-0349-y.

[9] Cariño RL. CMD Codes Repository—VPSC7b_gui [Internet]. [2009]. Available at: https://icme.hpc.msstate.edu/viewvc/CMD%20Codes%20Repository/VPSC7b_gui/ [Accessed: 2016-01-14].

[10] Tomé CN, Lebensohn RA. Manual for Code Visco-Plastic Self-Consistent (VPSC), Version 7b [Internet]. 2007. Available at: https://icme.hpc.msstate.edu/viewvc/CMD%20Codes%20Repository/VPSC7b_gui/trunk/doc/VPSC7b_manual.pdf?view=log [Accessed: 2016-01-14].

[11] Cariño RL. Multistage Fatigue Model Calibration Tool (v2) User Guide [Internet]. 2012. Available at: https://icme.hpc.msstate.edu/mediawiki/index.php/MSF_v2 [Accessed: 2016-01-14].

[12] Cariño RL. MEAM Parameter Calibration Tool (Version 4) User Guide [Internet]. 2015. Available at: https://icme.hpc.msstate.edu/mediawiki/index.php/MPC [Accessed: 2016-01-14].

[13] Sandia Corporation. LAMMPS Molecular Dynamics Simulator [Internet]. 1997 [Updated 2015]. Available at: http://LAMMPS.sandia.gov [Accessed: 2016-01-14].

[14] Sandia Corporation. pair_style MEAM command [Internet]. 2013. Available at: http://LAMMPS.sandia.gov/doc/pair_MEAM.html [Accessed: 2016-01-14].

[15] Baskes MI, Johnson RA. Modified embedded atom potentials for HCP metals. Modelling and Simulation in Materials Science and Engineering. 1994;2(1):147–163. DOI: 10.1088/0965-0393/2/1/011.

[16] Dassault Systèmes. Abaqus Analysis User's Manual [Internet]. 2007. Available at: http://www.egr.msu.edu/software/abaqus/Documentation/docs/v6.7/books/usb/default.htm?startat=pt05ch20s08abm59.html [Accessed: 2016-01-14].

[17] Horstemeyer MF, Cariño RL, Hammi Y, Solanki KN. CMD Codes Repository: DMG [Internet]. 2009 [Updated 2012]. Available at: https://icme.hpc.msstate.edu/viewvc/CMD%20Codes%20Repository/DMG/ [Accessed: 2016-01-14].

[18] Jordon JB, Horstemeyer MF, Solanki KN, Bernard JD, Berry J, Williams TN. Damage characterization and modeling of 7075-T651 aluminum plate. Materials Science and Engineering A. 2009;527(1):169–178. DOI: 10.1016/j.msea.2009.07.049.

[19] Bouvard JL, Cariño RL. Tutorial on the thermoplastic model calibration tool [Internet]. 2014. Available at: https://icme.hpc.msstate.edu/mediawiki/index.php/File:TPgui-1.2-Tutorial.zip [Accessed: 2016-01-14].

[20] Ma Q, Marin EB, Antonyraj A, Hammi Y, El Kadiri H, Wang PT, Horstemeyer MF. On predicting the channel die compression behavior of HCP magnesium AM30 using crystal plasticity FEM. In: Sillekens WH, Agnew SR, Neelameggham NR, Mathaudhu SN, editors. Magnesium Technology 2011. Hoboken: John Wiley & Sons, Inc.; 2011. pp. 583–587. DOI: 10.1002/9781118062029.ch107.

[21] McDowell DL, Gall K, Horstemeyer MF, Fan J. Microstructure-based fatigue modeling of cast A356-T6 alloy. Engineering Fracture Mechanics. 2003;70(1):49–80. DOI: 10.1016/S0013-7944(02)00021-8.

[22] Cariño RL, Kim S. MEAM Parameter Calibration Tool (Version 3) User Guide [Internet]. 2014 [Updated 2015]. Available at: https://icme.hpc.msstate.edu/mediawiki/index.php/MPCv3 [Accessed: 2016-01-14].

[23] Horstemeyer MF, Hughes JM, Sukhija N, Lawrimore WBII, Kim S, Carino RL, Baskes MI. Hierarchical bridging between ab initio and atomistic level computations: calibrating the modified embedded atom method (MEAM) potential (part A). Journal of Materials. 2015;67(1):143–147. DOI: 10.1007/s11837-014-1244-0.