Cloud-Computing Based Parameter Identification System – with Applications in Geotechnical Engineering

Y.M. Hsieh[1]

[1]Department of Construction Engineering, National Taiwan University of Science and Technology, No. 43, Sec. 4, Keelung Rd., Da-an District, Taipei, Taiwan; PH: 886-2-27301056; FAX 886-2-27376606; email: ymhsieh@mail.ntust.edu.tw

ABSTRACT

Parameter or system identification is a useful tool in civil engineering. They have wide applications across many disciplines. However, due to the lack of general-purpose tools for such analyses, researchers need to develop their own codes with duplicated efforts. Yet, it is sometime difficult to adopt other people's code due to different modeling methods used. Thus, there is a need for general-purpose tools to resolve problems of this kind.

This paper describes a general-purpose system for such purpose. The system has three features. First, it uses cloud-computing technology to provide large-scale capabilities. Second, the system uses a JavaScript engine for developing extensions. Third, it employs multi-objective particle swarm optimization for its applicability on wide range of problems. These features allow the system provide general capabilities to solve problems with similar needs. The design is validated and demonstrated by identifying constitutive model parameters in geotechnical engineering.

INTRODUCTION

System identification (SI) is the process of modeling an unknown system based on a set of input-outputs (Sirca and Adeli 2012). Hundreds of papers conducting SI since 1995 were reviewed by Sirca and Adeli (2012) in the field of structural engineering. SI also has several uses in geotechnical engineering. For example, finding constitutive model parameters by using observations in laboratory tests (Mulia 2012), supported excavation (Finno and Calvello 2005; Rechea et al. 2008), tunneling (Pichler et al. 2003), and in-situ testing (Zentar et al. 2001). SI for identifying model parameters is called parameter identification (PI). Many applications of SI or PI in open literatures have built their own programs to conduct the SI or PI process. No general-purpose parameter or system identification system seems to exist. Thus, many researchers have duplicate efforts on developing their own research codes. Significant efforts may still be needed when researchers are willing to share their codes. This is because different researches may want to use different modeling methods (e.g. finite element modeling vs. empirical simplified modeling). Therefore, there exists a need for general-purpose SI systems to avoid duplicated efforts among different researchers.

A general-purpose SI system is developed to answer the aforementioned demand. The system employs IT technologies developed for cloud computing to provide large-scale capabilities that can serve multiple users at the same time. This capability can also be used for trying out and comparing different modeling techniques when conducting SI. The system further incorporates V8 JavaScript engine developed by Google, Inc. This incorporation enables the system have easy extensibility and great flexibility, and will be explained in discussing the system design. Finally, the system implements one of the most general optimization technique known as multi-objective particle swarm optimization (Coello Coello and Lechuga 2002), or MO-PSO for short. The extensibility through V8 engine and general optimization capability given by MO-PSO makes the system developed general-purpose.

This paper is organized as the following. First, the general procedure for SI is described. Then, adapting PI in geotechnical engineering to the procedure described will be explained. This section also gives reasons to some design choices of the system developed. Next, the design and implementation of the system are revealed. The developed user interface to the system is then demonstrated. Finally, some concluding remarks are drawn basing on the experience for developing and using the system.

SYSTEM IDENTIFICATION

Figure 1 shows the general process for system identification. At beginning, unknown model parameters are generated using past experience or by random. Then, models such as finite element models with given parameters are computed to produce outputs. The outputs are then compared to observed behaviors, e.g. field measured displacements. The differences are then quantified by objective functions, which give rise to objective function values. If these values are small enough, then the system is identified, and unknown parameters are obtained. Otherwise, the optimizer will then try to minimize these values and guides updates to model parameters. Once parameters are updated by the optimizer, the process loops back until satisfactory.

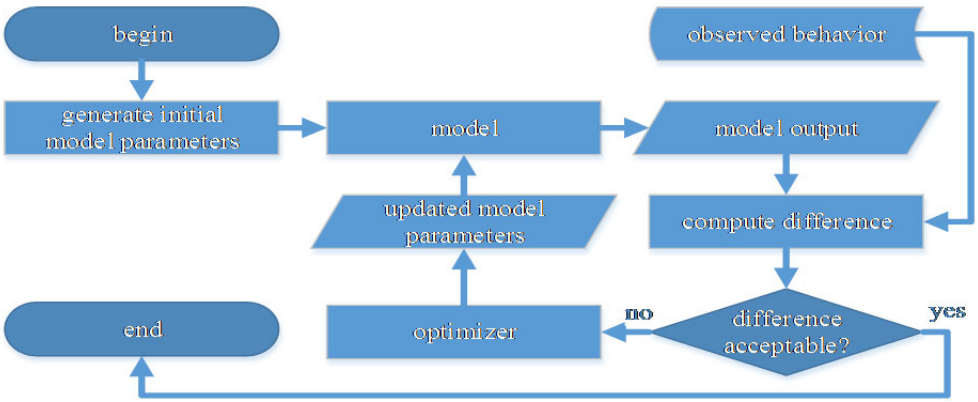

Figure 1. General procedure for system identification.

Parameter identification in geotechnical engineering. The SI process illustrated in Figure 1 can be applied for PI in geotechnical engineering. In particular, soil model parameters are focused in this paper. Because they are essential for accurate numerical models, and they can sometimes be difficult to obtain. For example, MCC (Roscoe and Burland 1968), S-Clay1 (Wheeler et al. 2003), Hardening-Soil (T. Schanz 1999), and MIT-S1 (Pestana and Whittle 1999) have respectively 5, 8, 9, and 16 parameters. These parameters often require more than one kind of laboratory tests to determine. For instance, parameters for MCC are often derived from oedometer tests for its main deformability parameters (λ, κ) and triaxial compression tests for the remaining parameters (p_0, M, ν). In other words, more than one observed behaviors are needed for successfully PI for such soil models. Therefore, the optimizer in Figure 1 needs to handle multiple objectives.

Many past studies used an averaged sum approach to convert multiple-objective problems into single-objective ones. However, such approach needs special attention on how to choose weights. For example, one observed behavior might be deformations of few millimeters while the other may be yield stress of several MPa. It is difficult to choose the right weight to combine them into one objective. Therefore, in author's opinion, if the PI process needs to be general, then the Pareto-optimal approach (Padhye 2009) is preferred for multi-objective optimization. Furthermore, if the SI or PI has only one objective, they can still be handled by the Pareto-optimal approach.

The model in Figure 1 can often be constructed by using finite element method (FEM). FEM is a general tool in structural and geotechnical engineering. It can account for complex constitutive models, geometries, loadings, boundary conditions, and even multi-physics phenomenon. It is therefore the ideal model choice for a general-purpose PI system. The drawback for using FEM, however, is it takes times for computation. Therefore, it is necessary for the developed PI system to use multiple processors and even multiple computers to help speed up the PI process.

SYSTEM DESIGN AND IMPLEMENTATION

In the following sections, the design and implementation of a general-purpose and cloud-computing based parametric identification system are described. The system is designed to serve multiple users, and cloud computing is used to serve their demands. The following sections describe 1) user input, 2) system design, and 3) user output of the system.

User input. For each parameter identification problem, users need to prepare a parameter identification task. This preparation includes a *task setting* and a *task archive*. The *task setting* is a file describing a parameter identification task. This file is written in JavaScript and will be processed by the *task executor* (described later in the system design). The file must define a JavaScript function config() and return a configuration object, explained in Table 1. From which, two efforts can be seen to make this system general purpose. The first effort is that each trial set of parameters can execute several analyses using different programs. These analyses are flexibly

defined by *cmdline* (defining how to execute an analysis) and *files* (defining files needed for analysis) in *analyses* objects. The second effort is one analysis can give rise to many objective function values. The *objectives* property is an array containing several objective objects. Each objective object has three main properties: *source*, *target*, and *objective*. The *source* property defines an associated analysis (in *from* property) and a method to extract analysis results (in remaining properties of *source*). The *target* property defines how the analysis result should look like by extracting data from a data file. Finally, the *objective* property defines the name of a JavaScript function (errorTXC in the example) written in the same file. This function computes an objective value basing on the extracted data from both the *source* and the *target*.

Once the *task setting* is prepared. Users need to create a *task archive***.** A *task archive* is a zip-compressed archive. The archive contains files needed for conducting analysis, and a *task-setting* file previously introduced. An example content of the archive is given in Table 2. Once the *task archive* is prepared. User need to upload the file the system to conduct the defined analysis.

Table 1. Parameter Identification Configuration Object (in JSON format).

Property	Purpose & Example
input	**Purpose**: Defile the file that contains variables to be identified. The developed system helps substitute parameters in the file with trial parameters suggested by the optimizer. **Example**: input: material.inc
parameters	**Purpose**: Define parameters and in the input file to be changed by optimizer. **Example**: parameters: ["$lambda", "$kappa", "$M", "$nuy"]
$lambda, $kappa, $M, $nuy	**Purpose**: Define ranges & initial value for each parameters to be identified. **Example**: $lambda: { range: [0.1, 0.2], initial: 0.15 }
analyses	**Purpose**: Define analyses to run for each trial set of parameters. It should be noted two analyses are defined in the following example, suggesting more than one analysis is allowed. **Example**: analyses: [{name: "txc-analysis", cmdline: "abaqus interactive job=txc",files: ["txc.inp"]}, {name:"txe-analysis", cmdline: "abaqus interactive job=txe", files: ["txe.inp"]}]
objectives	**Purpose**: Defines how to compute objective function values. It can contain several objects; each has a source property defining how to extract values from analyses and a target property defining the targets that should be matched. The following example shows only one of such object. **Example**: objectives: [{name: "txc-1",source: {from:"txc-analysis",output:"txc.dat",set:"set-1",fields: [2,3,4], rowsPerTable:1, steps:[1,2,3,4]}, target: {file:"lab-txc.txt", skip:1, columns: [0, 1], objective: "errorTXC" }}]

Note: In JSON, […] represents an array, and {…} represents an object. Inside objects are key-value pairs representing object's attribute and its value.

System architecture and implementation. The logical software architecture of the developed SI system is illustrated in Figure 2. It has two main components: cloud

computing, and parameter identification (PI). The cloud-computing component is responsible for providing cloud-computing benefits or characteristics. They are on demand self-service, network access, rapid elasticity, resource pooling, and measured service (Mell and Grance 2011). The PI component performs its task basing on user-defined task descriptions. These two components are discussed next.

Table 2. Sample task-archive file content.

Filename	Purpose
setting.json	The task-setting file describes how to conduct parameter identification.
material.inc	A file contains material definition for finite element analyses
txc.inp	A triaxial compression test FE-analysis input file, which includes material.inc for defining its material properties.
txe.inp	A triaxial extension test FE-analysis input file, which includes material.inc for defining its material properties.
lab-txc.dat	A text file contains triaxial compression test results from laboratory.
lab-txe.dat	A text file contains triaxial extension test results from laboratory.

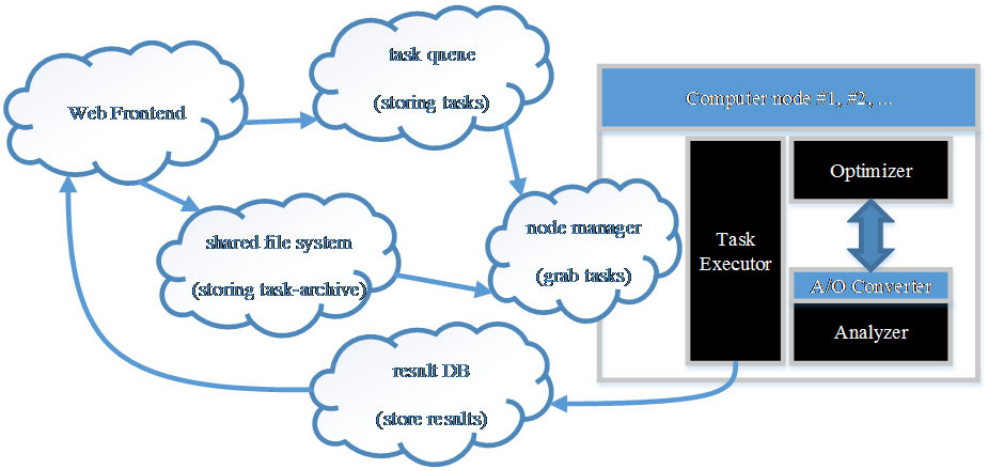

Figure 2. System architecture overview.

The cloud-computing component consists of five elements: Web frontends, a task queue, a shared file system, a result database, and node managers. They form the cloud-computing infrastructure for the developed system. Responsibilities of each element are summarized in Table 3. Users use PHP webpages hosted by Apache web server to upload their *task archive*. After it is uploaded, it is stored in the shared file system, and a new task is generated and stored in the task queue. The task queue is implemented as a collection in MongoDB, and the shared file system is implemented using the GridFS capability of MongoDB. The task queue will be polled periodically by node managers running on all computer nodes in a computer cluster or cloud-computing facility to see whether unexecuted tasks exist. Each unexecuted task will be fetched by one of the node manager and marked as grabbed in the task queue. Then, the task will be handed to the PI component for execution. During PI process, model parameters and objective values are stored to the result

database. The web frontend may pull results out from result database for display and visualization.

Table 3. Responsibilities of Cloud-Computing Elements.

Element	Responsibility	Implementation
Web frontends	User interface for inputs and outputs	PHP[1], Apache[2]
Task queue	Stores tasks and their status	MongoDB[3]
Shared file system	Store task descriptions	MongoDB
Result database	Store SI or PI results	MongoDB
Node managers	Get tasks from task queue and shared file system, and invokes task executor	PHP

[1]:http://www.php.net/ [2]:http://httpd.apache.org/ [3]:http://www.mongodb.org/

The PI component has four elements: task executor, optimizer, A/O converter, and analyzer. Their purposes are described in Table 4. The task executor, programmed in C^{++}, gets needed data from the node manager, and then it uses JavaScript engine V8 to parse and run the task-setting file to get the configuration object previously introduced. Accordingly, the task executor then sets up the optimizer and A/O converter. The task executor also establishes connections between optimizer, analyzer, and A/O converter. Then, the control is handed over to the optimizer to conduct the PI process previously introduced in Figure 1. During the process, optimizer invokes analyzer to run defined analyses in parallel using OpenMP. Once an analysis is ended, the optimizer invokes A/O converter to compute objective values. The optimizer implements MO-PSO, and data on particles' positions, objective function values, and Pareto front are all stored to the result database, implemented using MongoDB.

Table 4. Responsibilities of Parameter Identification Elements.

Element	Responsibility	Implementation
Task executor	Parse task-description file and configure other components accordingly.	C^{++}, V8[1]
Optimizer	Conduct objective-function minimization with the help of A/O convertor and Analyzer	C^{++}, OpenMP[2]
Analyzer	Conduct one analysis with parameter given by the optimizer.	C^{++}
A/O converter	Convert analysis results obtained from analyzer and converts into values of objective functions	C^{++}, V8

[1]:https://code.google.com/p/v8/ [2]: http://openmp.org/wp/

User Output. The outputs from PI, including tried parameters, objective function values, and Pareto front, are all stored in the result database. These data are presented to users through the Web frontends upon request. JQueryUI and Google chart API are used for presentation. Users can also obtain raw data in comma separated value file format to process them further in Excel.

SYSTEM DEMONSTRATION

Figure 3 shows the main screen for the system developed. The login box & task submission box is shown on the right-hand side of the main screen. Users must obtain a valid account before using the system. Once users have logged in, they can fill in any description in the description textbox, and then click on the "Choose File" button to choose their task archive. Once finished, they can click on the submit button to send their task archive to the system. Left-hand side of the main screen in Figure 3 is the task queue. Users can see how long the task has been in the queue and its status. Each task may have [V] button to view results, as in Figure 4; [L] button to see outputs from the task executor; and [A] button to download all data to local computer for further processing in other software, e.g. Excel.

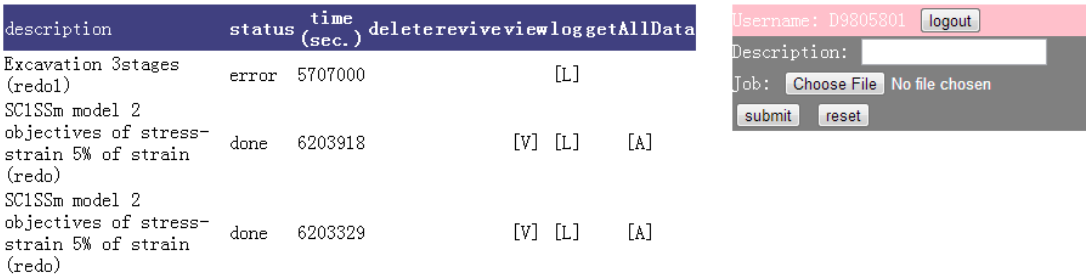

Figure 3. Main screen for users to submit their tasks.

Figure 4 shows the output from the system. It shows the number of iterations conducted for PI for chosen job. It uses a slider to choose the iteration number, from which the data are presented in the following. The system shows the number of particles in the Pareto front versus iteration number, allowing users to get some insight to the PI process. For a chosen iteration, the system displays its Pareto front solutions. Detailed parameters and objective values are presented in the tabular format.

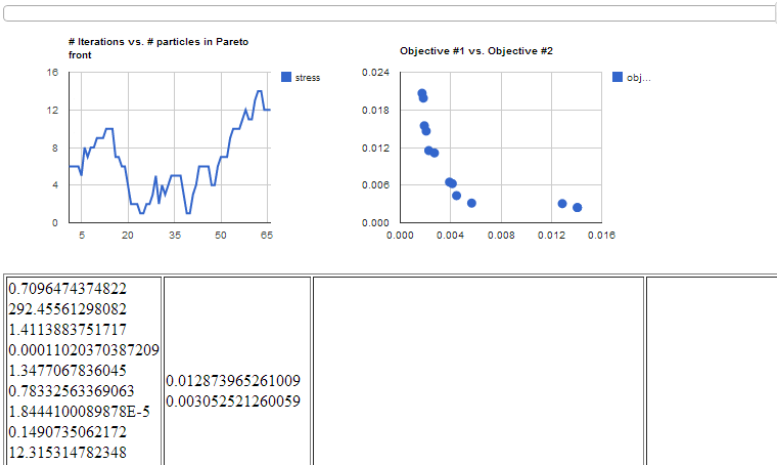

Figure 4. Simple outputs for showing iteration process & Pareto front.

CONCLUDING REMARKS

A PI or SI system is successfully implemented using the chosen software stack. In particular, using MongoDB greatly simplifies communication needs between different computers, making parallel and distributed computing easier than before. Interoperating between C++ and JavaScript also makes the system easy to extend. Further use of the system is planned, and collaboration using the system is welcomed.

ACKNOWLEDGEMENT

The author would like to acknowledge Huu-Phuoc Dang, a Ph.D candidate in the geotechnical engineering group in author's department, for his hard work preparing inputs of several cases for the system developed.

REFERENCES

Coello Coello, C. A., and Lechuga, M. S. "MOPSO: a proposal for multiple objective particle swarm optimization." *Proc., Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, 1051-1056.

Finno, R., and Calvello, M. (2005). "Supported Excavations: Observational Method and Inverse Modeling." *Journal of Geotechnical and Geoenvironmental Engineering*, 131(7), 826-836.

Mell, P., and Grance, T. (2011). "The NIST Definition of Cloud Computing." *Special Publication 800-145*, National Institute of Standards and Technology, U.S. Department of Commerce, 7.

Mulia, A. (2012). "Identification of Soil Constitutive Soil Model Parameters Using Multi-Objective Particle Swarming Optimization." Master thesis, National Taiwan University of Science and Technology, Taipei, Taiwan.

Padhye, N. (2009). "Comparison of archiving methods in multi-objectiveparticle swarm optimization (MOPSO): empirical study." *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, ACM, Montreal, Quebec, Canada, 1755-1756.

Pestana, J. M., and Whittle, A. J. (1999). "Formulation of a unified constitutive model for clays and sands." *International Journal for Numerical and Analytical Methods in Geomechanics*, 23(12), 1215-1243.

Pichler, B., Lackner, R., and Mang, H. A. (2003). "Back analysis of model parameters in geotechnical engineering by means of soft computing." *International Journal for Numerical Methods in Engineering*, 57(14), 1943-1978.

Rechea, C., Levasseur, S., and Finno, R. (2008). "Inverse analysis techniques for parameter identification in simulation of excavation support systems." *Computers and Geotechnics*, 35(3), 331-345.

Roscoe, K. H., and Burland, J. B. (1968). "On the generalized stress–strain behaviour of 'wet' clay." *Engineering Plasticity*, J. Heyman, and F. A. Leckie, eds., Cambridge University Press, 535-609.

Sirca, G. F., and Adeli, H. (2012). "System identification in structural engineering." *Scientia Iranica*, 19(6), 1355-1364.

T. Schanz, P. A. V., P.G. Bonnier (1999). "The hardening soil model: Formulation and verification." *Beyond 2000 in Computational Geotechnics*, Balkema, Rotterdam.

Wheeler, S. J., Näätänen, A., Karstunen, M., and Lojander, M. (2003). "An anisotropic elastoplastic model for soft clays." *Canadian Geotechnical Journal*, 40(2), 403-418.

Zentar, R., Hicher, P. Y., and Moulin, G. (2001). "Identification of soil parameters by inverse analysis." *Computers and Geotechnics*, 28(2), 129-144.