IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 7, 2018, accepted December 30, 2018, date of publication February 8, 2019, date of current version March 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2897486

A Regression-Based Collaborative Filtering Recommendation Approach to Time-Stepping Multi-Solver Co-Simulation

JIAXIN ZHAO[1], HONGWEI WANG[2], AND HEMING ZHANG[1]
[1]Department of Automation, Tsinghua University, Beijing 100084, China
[2]ZJU-UIUC Institute, Zhejiang University, Haining 314400, China

Corresponding authors: Hongwei Wang (hongweiwang@intl.zju.edu.cn) and Heming Zhang (hmz@mail.tsinghua.edu.cn)

This work was supported in part by the National Key R&D Program of China under Grant 2018YFB1701602, in part by the State Key Laboratory of Intelligent Manufacturing Systems Technology under Grant QYYE1601, and in part by the National Natural Science Foundation of China under Grant 61374163. This work was also supported in part by the Zhejiang University/University of Illinois at Urbana–Champaign Institute, and was led by Principal Supervisor Prof. Hongwei Wang.

ABSTRACT The ever-increasing application of modeling and simulation to the development of complex engineering systems has made co-simulation indispensable to the handling of coupled multi-domain models. The mechanism for controlling communication between multiple solvers holds the key to co-simulation performance and is regarded as one of the most challenging parts in co-simulation as a lot of tradeoffs need to be made in terms of stability, accuracy, and efficiency. As such, a holistic and dynamic approach is required, which has not been addressed by this paper that has a focus on either tailored problem with a specific numerical analysis scheme or software platforms for implementing data exchange. This paper precisely aims to address this gap by developing a knowledge-based approach to streamlining the co-simulation process. Specifically, a regression-based collaborative filtering approach is developed to recommend suitable ordinary differential equation solvers for individual simulators according to the specific engineering characteristics and historical simulation data. On this basis, the theoretical analysis of the stability region and truncation error is conducted to provide guidance on controlling time stepping of individual simulators using a Jacobi communication scheme. This approach has been evaluated in several computational experiments, in which the advantages of the proposed approach are demonstrated. First, the recommendation algorithm is reliable in making suggestions on viable solvers during simulation run time, especially when only sparse historical datasets are available. Second, the time-stepping scheme noticeably improves the computational efficacy owing to it having no dependence on the initial step-size choice, which is a more eminent advantage for high-fidelity co-simulation problems.

INDEX TERMS Co-simulation, regression-based collaborative filtering, ODE solver recommendation, simulator selection, step-size control.

I. INTRODUCTION

Multidisciplinary analysis has become an essential part of the design process of complex engineering systems, e.g. aircrafts and automobiles, which involves consideration of multiple coupling factors between subsystems in terms of function and structure. As the ever-increasing applications of domain-specific simulation tools, co-simulation has become indispensable to the handling of coupled models created for different domains. In the context of this research, Multi-solver Co-Simulation (MCS) refers to the process whereby the requirements for the simulation task of a complex system are transformed into multiple models as well as a scheme for solving these models in parallel using various numerical solvers. A general MCS process mainly consists of five parts [1]:

1) Identification of requirements such as assumptions of modeling, setting of simulation (e.g. loads to be applied), simulation performance (e.g. accuracy and speed), etc.;

2) Mathematical modeling based on detailed analysis of the physical features (e.g. boundary condition) and creation of simulation models for individual subsystems

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

3) Simulation decomposition and modularity analysis – this part is optional is if the unsolved simulation problem is decoupled;

4) Co-simulation scheme determination, e.g. sub-solver selection and communication mechanism selection;

5) Simulation results post-processing such as visualization, validation and efficiency analysis.

Amongst these five parts, elaboration of a co-simulation scheme is of critical importance to simulation performance in terms of both accuracy and reliability, and yet numerous challenges remain to be addressed. The research reported in this paper aims to develop an effective and efficient co-simulation scheme and its key parts are highlighted in red in the MCS process flowchart in Fig. 1.

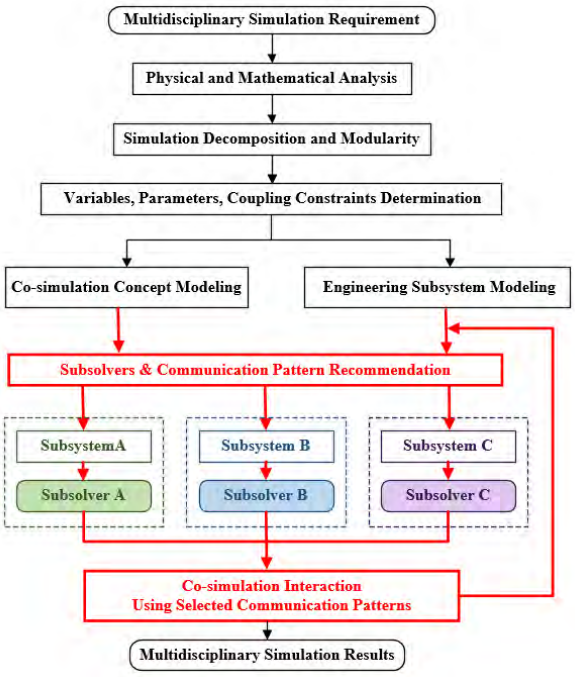

FIGURE 1. A flowchart of the MCS process.

Specifically, the first issue that needs to be addressed is the selection of numerical schemes for different sub-solvers (i.e. the solver for a particular subsystem model) during MCS execution. The main reasons for choosing different numerical schemes for these subsystem models are two-fold. First, each model is domain-specific and has different mathematical features. As such, a suitable solver needs to be selected to address these features. A few examples can be given to explain this: a solver using an implicit scheme is generally more stable than an explicit solver with the same level of accuracy despite being more computationally expensive [2]; a solver with coarse mesh is more efficient than the one with refined mesh but accuracy and stability is compromised [3]; a solver embedded with a multigrid method is proposed to solve a large-sparse matrix, which is not suitable for a small-dense matrix [4]. Second, even if the subsystems in a MCS task have similar mathematical formulations, using the same solver may still incur computational waste in terms of both memory and time (e.g. the case in Fig. 2) due to different dataset sizes, termination criteria and other simulation requirements. As a consequence, numerical scheme recommendation is greatly needed to improve simulation performance.

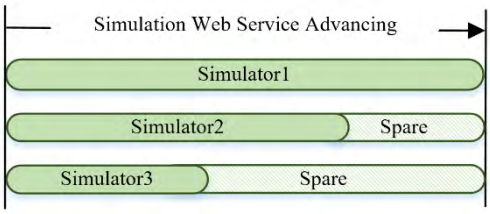

FIGURE 2. Waiting caused by each simulator using the same solver.

The second issue raised is then to develop an effective and efficient algorithm for recommending suitable numerical schemes for different MCS solvers. Literature search has shown that little work has been done on this. The algorithms for making recommendation typically include Collaborative Filtering (CF), personality-based (content-based) approach, demographic techniques, and a combination of these methods (i.e. hybrid recommender system) [5], and in practice these have been used in large online shopping companies and movie rating websites [6]. Unlike the E-commerce applications on the Internet with large datasets of items and users, the MCS solver recommendation issue faces the challenge of data sparsity, meaning that it is difficult to build a large similarity matrix due to the lack of historical usage and rating information [7]. To address this challenge, this paper proposes a novel regression-based CF algorithm that can be used for numerical schemes recommendation in MCS.

The rest of this paper details the methods for addressing the above two issues to underpin an adaptive time-stepping MCS solution using a regression-based CF recommendation scheme. Specifically, a literature review of MCS treatments and CF methods is conducted in Section 2. Section 3 describes the proposed approach with detailed information about its three main steps. Section 4 provides a case study to explain the application of the proposed approach as well as to evaluate its performance. In Section 5, the proposed approach is further evaluated by conducting additional experiments based on a dataset with different simulation problems and solvers. The conclusions of this paper are drawn in Section 6.

II. LITERATURE REVIEW

In this section, related work on MCS treatment and CF recommendation methods is reviewed and discussed.

A. MCS TREATMENT

Table 1 lists different MCS treatment methods, in terms of both computer software development and numerical analysis

TABLE 1. Summary of current mcs treatments.

Perspective	Categories			Simulation Performance			Multidisciplinary Application
				Stability	Accuracy	Speed	
Computer: Software Development	Tailored Problem (In-House Solver /Code)			High	High	High	• Fluid-Structure(CFD) [8] • Multibody System(Finite Element) [9] • Atmospheric Chemistry[10]
	General Purpose (Public Scheme / Platform)			Low	Low	Low	• High Level Architecture (HLA)[10] • Functional Mock-up Interface[11] • Agent-based Dynamic Model[12]
				Reuse Capability: High			
Mathematical: Numerical Analysis	Subsystem Solver*1	Explicit		Low	Low	High	• System with Sparse Matrix[13]
		Implicit		High	High	Low	• Stiff System[14]
		Semi-Implicit		High	Medium	Medium	• System with Dense Matrix [15]
	Communication Pattern	Fixed Time-Stepping*2	Serial	High	High	Low	• Gauss-Seidel / Serial Stagger[16]
			Parallel	Low	Low	High	• Jacobi Scheme[17]
		Adaptive Time-Stepping		Uncertain	Uncertain	High	• Optimal Strategy[18] • Bisection Strategy[19]

*1: The simulation performance comparison of subsystem solvers is based on the same order schemes
*2: The simulation performance comparison of fixed time-stepping communication patterns is based on the same step-size and simulation time interval

issues, and on this basis compares different co-simulation approaches.

From a computer-aided MCS development perspective, there are mainly two treatments for MCS. The first one focuses on tailored computational solutions for specific engineering problems. This treatment method requires every detailed information about the simulation problem. Engineers and system analysts need to establish the mathematical formulation for each subsystem, analyze the communication pattern theoretically and develop bespoke code for continuously updating information obtained from other subsystems. This treatment method is generally used for the circumstances in which subsystems represent different areas of physics. For instance, a fluid-structure problem is solved by Farhat and Lesoinne [20] using staggered pattern based on Aero-F and Aero-S solvers; a thermomechanical problem is treated by Armero and Simo [21] using a fractional step method based on a two phase operator; distributed simulation of a multibody dynamic system is implemented by Wang *et al.* [22] using a gluing algorithm which couples different component models in a plug-and–play manner.

On the other hand, another category of MCS treatment methods is focused on the investigation of general-purpose communication patterns in particular when little knowledge is available about the subsystems. The main advantage of this technique, compared with the tailored solution, is that it can provide general guideline and a reusable framework for combining well-established software packages such as Matlab, Modelica, Adams, Nastran, Ansys, Abaqus, Cosmos and Altair. The computational infrastructure widely used in distributed computing provides useful solutions for such a scheme, which underpins data exchange between computational models while hiding the technical details of distributed communication from end users. The Functional Mock-up Interface (FMI) [11] is a tool-independent standard using a combination of an executable binary xml-files and embedded complied C-code, which can support both model exchange and co-simulation of dynamic models. The Common Object Request Broker Architecture (CORBA) [23] enables computational models using different programming languages to work together. The High Level Architecture (HLA) [24] emphasizes distributed co-simulation management of discrete event systems by providing platform-independent rules, interfaces and data models. Moreover, Web Services technologies developed in cloud-computing [25] can also enhance the interoperability of applications through facilitating remote communication [26]. Another Web-based solution applied to system integration in complex engineering system development is multi-agent technology [27].

In addition to industrial standard and computational infrastructure, some general co-simulation approaches can also be found in numerical analysis literature. The engineering co-simulation systems are usually modeled according to physical and dynamic laws, which are described in the form of Partial Differential Equations (PDEs), Differential Algebraic Equations (DAEs) and Ordinary Differential Equations (ODEs). These differential equations usually possess specific characteristics with regards to their variables, coefficients and boundary conditions, including linearity, degree of order and stiffness. In order to solve a tailored problem using computational methods, various numerical integration schemes and techniques have been developed to make a trade-off between different performance indicators. Usually explicit, implicit and semi-implicit methods are adopted as three subdivisions of computational methods based on different simulation performance requirements in terms of stability, accuracy and speed. As a Jacobian-free method, the explicit approach is the most effective one to be implemented, especially in the cases that require data exchange between commercial packages which allows limited access to solver codes. Meanwhile,

the disadvantage of the explicit scheme is its numerical instability. Compared with the explicit approach with the same order, the semi-implicit approach can achieve better stability with Jacobian-based calculation. Schweizer and Lu [28] proposed a semi-implicit scheme to tackle mechanical solver coupling using small dimension Jocobian matrix, which still need improvement in terms of accuracy and stability through performing a complete iteration operation. Amongst all the methods, the implicit approach is the most stable with the cost of repetitive iteration for solving algebraic equations at each simulation step. For instance, Sicklinger [29] introduces an interface Jacobian-based algorithm to solve co-simulation scenarios involving an arbitrary number of fields and signals which overcomes present stability issues but incurs additional computation time. Based on the above discussion, these approaches still face accuracy and efficiency challenges and in-depth analysis is needed for their effective application to MCS problems.

B. CF RECOMMENDATION METHODS

In general sense, Collaborative Filtering (CF) is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints and data sources [30]. In the context of this paper, CF is used as an efficient machine learning technique to extract useful information based on historical dataset so that viable co-simulation mechanisms with the best performance can be recommended for sub-solvers.

Early generation CF systems usually utilize rating data to calculate the similarity or weight between users or items and make predictions, which is called memory-based CF method [31]. However, a significant limitation of memory-based CF method is that the similarity matrix is based on common items and thus is unreliable when data are very sparse. To overcome this shortcoming, many model-based CF methods have been proposed, such as the regression-based CF model [32], the Bayesian belief nets CF model [33], the clustering CF model [34], the latent semantic CF model [35], and the Markov-decision-process CF model [36]. These approaches use machine learning algorithms to build a prediction model based on historical information and can efficiently overcome data sparsity by satisfying some general evaluation rules.

Moreover, some dimensionality reduction techniques are employed to remedy sparsity in CF systems, which are termed Matrix Factorization (MF) [37]. For instance, Singular Value Decomposition (SVD) [38] can remove insignificant users or items to reduce the dimensionalities of user-items matrix directly. Principal Component Analysis (PCA) is usually regarded as a SVD modification method and Probabilistic Matrix Factorization (PMF) [39] is also a well-established method for recommendation system in practice.

Among all the model-based CF methods, the regression-based CF proves to have advantages in making predictions for numerical values in recommender systems. For example, Vucetic and Obradovic [32] proposed a regression-based CF approach which combines simple linear models to provide rating predictions for an active user. Ordinary least squares is used in the approach to estimate the parameters of regression function. The good performance in addressing the sparsity issue is demonstrated in experiments using many benchmark datasets. In this research, the regression-based CF is employed and adapted in the knowledge-based recommendation approach due to its explicit mathematical representation, easy implementation and efficient addressing of data sparsity.

III. THE PROPOSED APPROACH

A general MCS scheme is first given to explain the application of the proposed approach, which contains several partitioned systems. Each of them can be expressed as Ordinary Differential Equations (ODEs) and will complete the time-stepping numerical processes in a cooperative way. Then, based on the different characteristics and historical dataset of the ODEs, a suitable ODE solver is recommended by the regression-based CF algorithm. After this, recommendation results are used by the MCS process. The co-simulation pattern is determined in the last part of this process. A flowchart of the proposed approach is shown in Fig. 3.

A. STEP1: MCS FORMULATION

To explain the general idea of the time-stepping co-simulation scheme, a monolithic system represented by a set of first order ODEs with initial values is considered. It is semi-discretized from PDEs, as shown in Equation (1).

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \cdots \\ \dot{y}_i \\ \cdots \\ \dot{y}_m \end{pmatrix} = \begin{pmatrix} f_1(y_1, y_2, ..., y_i, ..., y_m) \\ f_2(y_1, y_2, ..., y_i, ..., y_m) \\ \cdots \\ f_i(y_1, y_2, ..., y_i, ..., y_m) \\ \cdots \\ f_m(y_1, y_2, ..., y_i, ..., y_m) \end{pmatrix},$$
$$\begin{pmatrix} y_1(t_0) \\ y_2(t_0) \\ \cdots \\ y_i(t_0) \\ \cdots \\ y_m(t_0) \end{pmatrix} = \begin{pmatrix} y_{10} \\ y_{20} \\ \cdots \\ y_{i0} \\ \cdots \\ y_{m0} \end{pmatrix} \tag{1}$$

In the equation, $i = 1, 2, ..., m$ represents the number of first order ODE for the monolithic representation. In terms of derivative notation, Newton's notation for differentiation is adopted, which places a dot over the function name to represent a time derivative and is thus also called the dot notation. For instance, $\dot{y} := dy/dt$ is the first derivative of y with respect to t. t_0 is the initial time and y_{i0} is the initial value. Actually, this description does not lose high order generality, because a high order ordinary differential scheme can always be rewritten as a system of first order ODEs. At the same time, PDEs can be numerically solved by time-stepping ODEs in a certain spatial dimension using this description. Furthermore, it is noteworthy that in real physical applications (e.g. the fluid-structure co-simulation

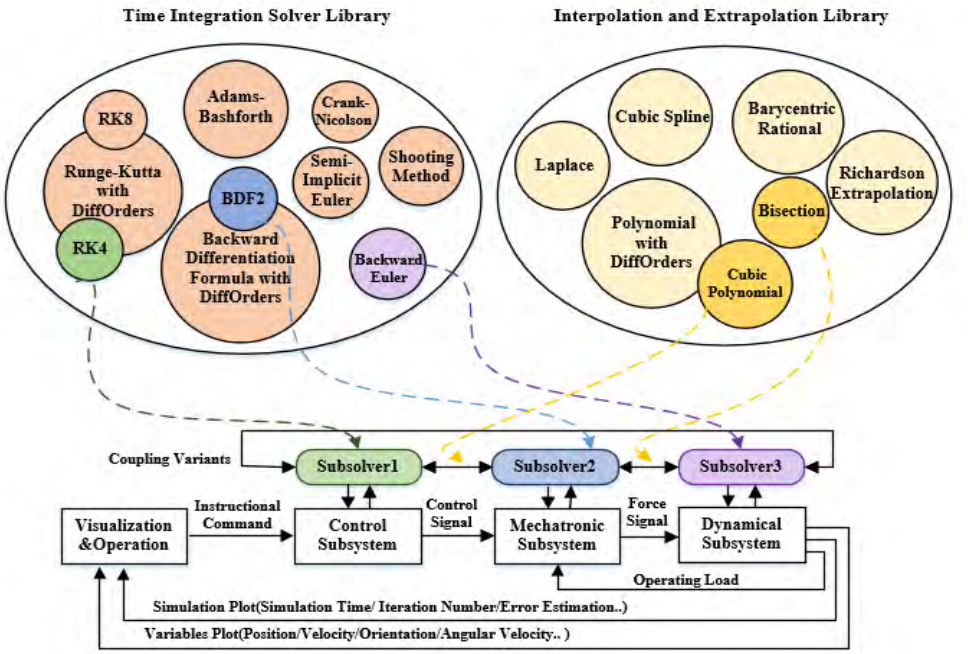

FIGURE 3. The regression-based collaborative filtering recommendation approach.

for airfoil), the monolithic representation usually does not exist. The monolithic form appearing in this paper is used for numerical analysis and experiment validation.

Next, the representation of partition systems is considered. Different from many other studies in which two subsystems are normally used [40], this research extends the coupling relationship to three subsystems (named A, B, and C). Hence, the communication pattern analysis can more likely be utilized for arbitrary subsystems. Specifically, Subsystem A ($i = 1, 2, ..., k$) has an ODEs representation shown in Equation (2)

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dots \\ \dot{y}_k \end{pmatrix} = \begin{pmatrix} f_1(y_1, y_2, ..., y_i, ..., y_m) \\ f_2(y_1, y_2, ..., y_i, ..., y_m) \\ \dots \\ f_k(y_1, y_2, ..., y_i, ..., y_m) \end{pmatrix},$$
$$\begin{pmatrix} y_1(t_0) \\ y_2(t_0) \\ \dots \\ y_k(t_0) \end{pmatrix} = \begin{pmatrix} y_{10} \\ y_{20} \\ \dots \\ y_{k0} \end{pmatrix} \tag{2}$$

Subsystem B ($i = 1, ..., p$) and Subsystem C ($i = 1, 2, ..., q$) have similar representation of ODEs

Usually $k < m, p < m, q < m$ and $k + p + q = m$. To simplify the following notions, we use vector to represent the undivided system in Equation (1) as follows

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0, \quad \mathbf{y} \in R^m \tag{3}$$

Similarly, each subsystem has the following form

$$\dot{\mathbf{y}}_A = \mathbf{f}(\mathbf{y}), \quad \mathbf{y}_A(t_0) = \mathbf{y}_{A0}, \quad \mathbf{y}_A \in R^k \tag{4}$$
$$\dot{\mathbf{y}}_B = \mathbf{f}(\mathbf{y}), \quad \mathbf{y}_B(t_0) = \mathbf{y}_{B0}, \quad \mathbf{y}_B \in R^p \tag{5}$$
$$\dot{\mathbf{y}}_C = \mathbf{f}(\mathbf{y}), \quad \mathbf{y}_C(t_0) = \mathbf{y}_{C0}, \quad \mathbf{y}_C \in R^q \tag{6}$$

To provide a better understanding of the problem formulation, the coupling relationship among the multidisciplinary systems is presented in Fig.4, which consists of three different functional subsystems linked by state vectors.

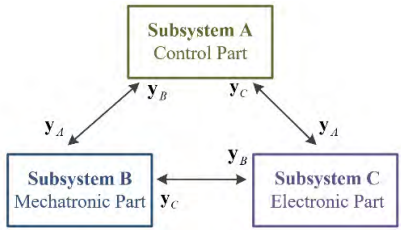

FIGURE 4. The coupling representation between three subsystems.

B. STEP2: SUB-SOLVER RECOMMENDATION

In order to recommend a suitable ODE solver for different subsystems, the history statistics of solvers simulation performance are important reference for building the rating list such as the one shown in Table2. The solvers Backward Euler [41], 2nd-order Backward Differentiation Formula (BDF2) [42], 4th-order Runge-Kutta (RK4) [43] and 4th-order Adams-Bashforth(AB4) [44] are used in Table 2.

Practically, in many ODE solver libraries, such as RADAU5 [45], DASPK [46], solvers are tested to collect simulation performance data using different ODE sets. In particular, DETSET [47] is a program which applies six ODE solvers to five categories of problems to monitor a solver's computational cost and reliability. The key results have been collected including FCN CALLS (the number of function calls required for the problem),

TABLE 2. An example of solver-subsystem matrix.

ODE Solver Type	Subsystem A	Subsystem B	Subsystem C
Backward Euler	Rating(1,1)	Unknown	Rating(1,4)
BDF2	Unknown	Rating(2,2)	Unknown
RK4	Unknown	Rating(3,2)	Rating(3,4)
AB4	Rating(4,1)	Unknown	Unknown

NO. OF STEPS (the number of steps used), PERCENT DECEIVED (the percentage of steps in which the local error exceeded the tolerance), MAXIMUM ERROR (maximum local truncation error per unit step). Based on this dataset, a rating score can be obtained using Equation (7) as follows.

$$r = \left(\frac{FCN \times NO.}{1 - 0.01 \times PER} \times \exp(MAX) \right)^{-1} \tag{7}$$

In the equation, FCN, NO., PER, MAX are abbreviations for FCN CALLS, NO. OF STEPS, PERCENT DECEIVED, MAXIMUM ERROR, respectively.

Then r can be scaling between 1-5 using Equation (8) as follows.

$$r_{norm} = 1 + \frac{4 \times (r_{\max} - r_{\min})}{r - r_{\min}} \tag{8}$$

The regression-based CF algorithm can be expressed as follows:

Step 1: Initialization of features $x^{(1)}, ..., x^{(n_m)}$ and parameters $\theta^{(1)}, ..., \theta^{(n_u)}$ using random values with standard normal distribution.

Step 2: Modeling of the cost function based on multivariable regression with regularization of features and parameters simultaneously

Step 3: Minimization of cost function to obtain $x^{(1)}, ..., x^{(n_m)}$ and $\theta^{(1)}, ..., \theta^{(n_u)}$ simultaneously using advanced optimization method.

Step 4: Measurement of Euclidean distance for each subsystem with (learned) parameters to compute similarity, predict a rating list of ODE solvers and recommend the solver with highest rating score.

For a subsystem with learned parameters, implementation of this algorithm is provided in Section IV.C where a detailed example is given.

C. STEP3: CO-SIMULATION PATTERN DETERMINATION

With the mathematical description of the three-subsystem problem, the next step is to solve these ODEs in a cooperative manner. Since iterative computation is the most efficient numerical measure for time-stepping simulation, one or two rounds of co-simulation running round can be used for explanation and this procedure can be executed repeatedly in a specified time interval. In a single co-simulation running round from t_n to t_{n+1}, each sub-solvers embedded in a subsystem calculate its own ODEs separately within time interval t, and then they exchange numerical calculation results with each other using a means determined by MCS communication pattern. In terms of communication pattern, there are mainly two categories, namely fixed step- size interaction and adaptive step-size interaction. Specifically, the former includes the Jacobi pattern (the parallel one) and the Gauss-Seidel pattern (the serial one). Fig. 5 shows the detailed differences of the Jacobi pattern and the Gauss-Seidel pattern in the co-simulation problem with three sub-solvers described above. The Jacobi pattern exchanges data simultaneously at the end of each time interval, while the Gauss-Seidel pattern exchanges data three times in a complete time interval. In this way, the Jacobi pattern is more computationally efficient but the Gauss-Seidel pattern is more accurate since Sub-solver 2 can utilize the updated data at the current time step from Sub-solver 1 while Sub-solver 3 can use the updated data from both Sub-solver 2 and Sub-solver 3.

However, each subsystem usually has a specific engineering function and may use different sub-solvers according to its engineering characteristics. In this way, a sub-solver's information can only be obtained at the end of each complete time interval. Thus, the Guass-Seidel pattern is no longer available for a co-simulation problem if each sub-solver uses a different numerical scheme, while the Jacobi communication pattern is more flexible. On the other hand, since the sub-solvers have different numerical integration schemes related to its order of accuracy, using a fixed step-size communication pattern is less efficient than using an adaptive step-size.

With these considerations, an adaptive step-size Jacobi communication pattern is proposed in this research for running the simulation with the three sub-solvers described above. The integration and interaction in a certain time interval is depicted in Fig.6. In order to accurately control step-size, estimating the Local Truncation Error (LTE) is required for each sub-solver. The uniqueness of certain derivative scheme is utilized in each sub-solver to achieve better step-size control in the section to follow.

IV. CASE STUDY

This section uses a case study to detail the proposed regression-based CF recommendation approach for time-stepping MCS and demonstrate how it can be used to solve a coupled three ODEs co-simulation problem.

A. PROBLEM DESCRIPTION

In order to demonstrate a derivative scheme with three specified sub-solvers, an oscillation mechanics model with dynamical coupling is chosen in this study, as shown in Fig. 7 [48]. This linear structure problem has some advantages as an demonstration model. First, it contains three sets of second-order ODEs and can be represented as a vector form. Second, it has both a monolithic form and a partitioned form, which makes it convenient to compare the proposed co-simulation algorithm with monolithic simulation results. Last, it can end up with a damped state which is easy to verify stability of the proposed algorithm.

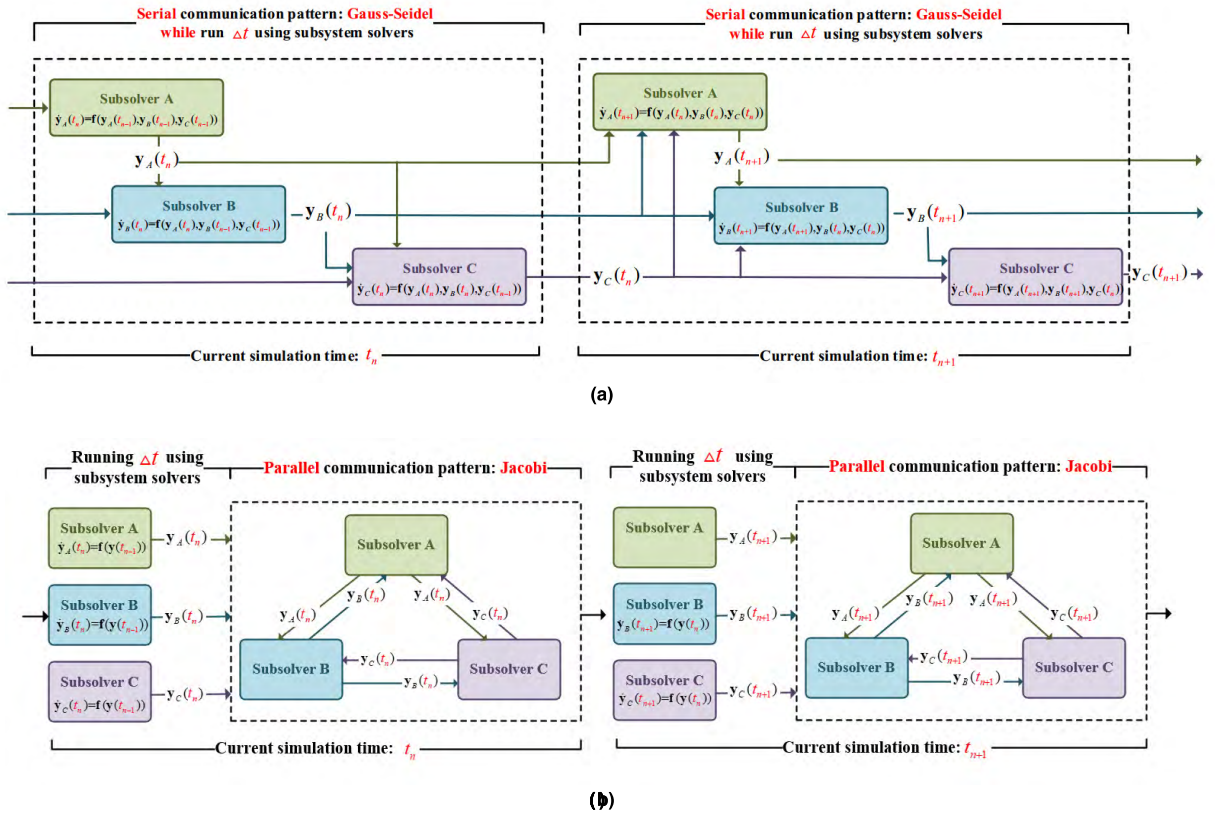

FIGURE 5. Fixed step-size communication pattern. (Note: without loss of generality, the explicit form is used in each solver as a demonstration.) (a) Serial communication pattern. (b) Parallel communication Pattern.

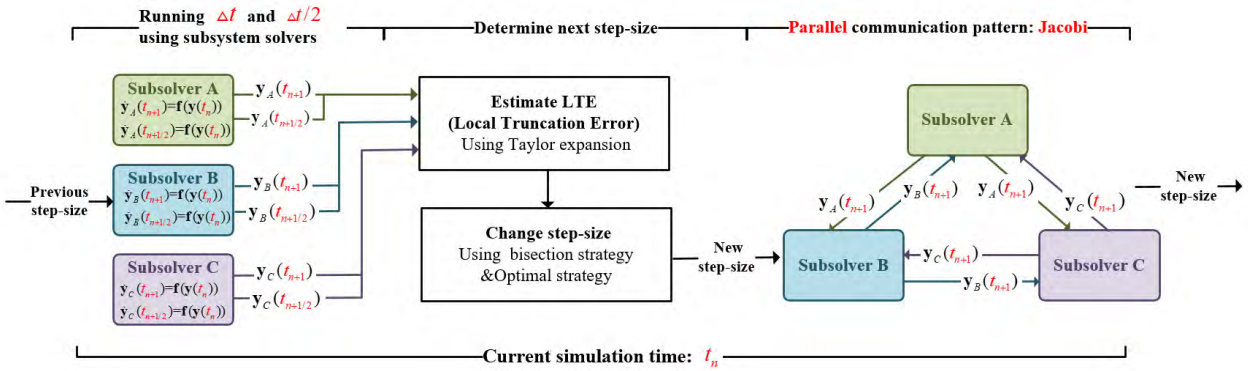

FIGURE 6. Adaptive step-size Jacobi communication pattern.

This oscillation mechanics model has the mathematical description as a monolithic system

$$\begin{cases} \ddot{u} + \dfrac{d_1}{m_1}\dot{u} + \dfrac{k}{m}u + \dfrac{k}{m}(u - v) = 0 \\ \ddot{v} + \dfrac{d_2}{m_2}(\dot{v} - \dot{w}) + \dfrac{k_2}{m_2}(v - u) = 0 \\ \ddot{w} + \dfrac{d_2}{m_3}(\dot{w} - \dot{v}) + \dfrac{k_3}{m_3}w = 0 \end{cases} \tag{9}$$

Equation (9) can also be rearranged in a matrix form including six first order ODEs as follows

$$\dot{\mathbf{y}} = F\mathbf{y}, \quad \mathbf{y} \in R^6 \tag{10}$$

where $\mathbf{y} = \begin{bmatrix} u & \dot{u} & v & \dot{v} & w & \dot{w} \end{bmatrix}^T$ is displacement vector and

$$F = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -\dfrac{k_1 + k_2}{m_1} & -\dfrac{d_1}{m_1} & \dfrac{k_2}{m_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \dfrac{k_2}{m_2} & 0 & -\dfrac{k_2}{m_2} & -\dfrac{d_2}{m_2} & 0 & \dfrac{d_2}{m_2} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \dfrac{d_2}{m_3} & -\dfrac{k_3}{m_3} & -\dfrac{d_2}{m_3} \end{bmatrix}$$

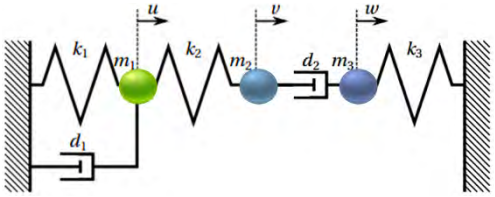

FIGURE 7. An oscillation mechanics model with dynamical coupling.

This system can also be partitioned into three parts and each subsystem describes the dynamical motion of a ball in Fig. 7. The subsystem 1 can be reformulated as

$$\dot{\mathbf{y}}_A = F_A \mathbf{y}_A + F_{AB} \mathbf{y}_B \tag{11a}$$

$$\dot{\mathbf{y}}_B = F_B \mathbf{y}_B + F_{BA} \mathbf{y}_A + F_{BC} \mathbf{y}_C \tag{11b}$$

$$\dot{\mathbf{y}}_C = F_C \mathbf{y}_C + F_{CB} \mathbf{y}_B \tag{11c}$$

where $\mathbf{y}_A = \begin{bmatrix} u & \dot{u} \end{bmatrix}^T$, $\mathbf{y}_B = \begin{bmatrix} v & \dot{v} \end{bmatrix}^T$, $\mathbf{y}_C = \begin{bmatrix} w & \dot{w} \end{bmatrix}^T$ and $F_A, F_B, F_C, F_{AB}, F_{BA}, F_{BC}, F_{CB}$ have a following relationship with F.$F_{AB}, F_{BA}, F_{BC}, F_{CB}$ are the coupling part in this simulation problem.

$$F = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{k_1+k_2}{m_1} & -\frac{d_1}{m_1} & \frac{k_2}{m_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{k_2}{m_2} & 0 & -\frac{k_2}{m_2} & -\frac{d_2}{m_2} & \frac{d_2}{m_2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{d_2}{m_3} & 0 & -\frac{k_3}{m_3} & -\frac{d_2}{m_3} \end{bmatrix}$$

B. RECOMMENDATION ALGORITHM FOR THREE SUB-SOLVERS USING REGRESSION-BASED CF

First, data preprocessing includes the following three procedures,

1) Building rating matrix $r \in R^{n\times 3}$, $r^{(i,j)}$ is the rating score by user subsystem j on solver i obtained by Equations (7) and (8).

2) Building $F \in R^{n\times 3}$ filtering matrix with sparsity around 50%, which only contains random values 0 or 1 and has the same size as the rating matrix.

3) Building filtered rating matrix $y \in R^{n\times 3}$, $y^{(i,j)}$(defined only if $F(i,j) = 1$).

The regression-based cost function with regularization is

$$J(x^{(1)}, ..., x^{(n_m)}, \theta^{(1)}, ..., \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):F(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2 \tag{12}$$

The minimization objective is as follows

$$x^{(1)}*, ..., x^{(n_m)}*, \theta^{(1)}*, ..., \theta^{(n_u)}* = \arg \min_{\substack{x^{(1)},...,x^{(n_m)} \\ \theta^{(1)},...,\theta^{(n_u)}}} J(x^{(1)}, ..., x^{(n_m)}, \theta^{(1)}, ..., \theta^{(n_u)}) \tag{13}$$

After formation of the cost function, advanced optimization method is used at every $j = 1, ..., n_u, i = 1, ..., n_m$ to obtain optimal feature and parameters in an iterative way. The regularized gradient which is used for optimization can be expressed as follows

$$\frac{\partial J}{\partial x_k^{(i)}} = \sum_{j:F(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta_k^{(j)} + \lambda x_k^{(i)} \tag{14}$$

$$\frac{\partial J}{\partial \theta_k^{(j)}} = \sum_{i:F(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(j)} + \lambda \theta_k^{(i)} \tag{15}$$

The detailed algorithm using the terms and equations mentioned above is described using the following procedure.

Algorithm 1 Regression-Based CF Recommendation for ODE Multi-Solver

Data preprocessing: through 1) – 3)
Input: filtered rating matrix $y \in R^{n\times 3}$, maximum gradient descent iteration MaxIter, regularization parameter λ, numbers of features,
Output: optimized features and parameters $x^{(1)}*, ..., x^{(n_m)}*, \theta^{(1)}*, ..., \theta^{(n_u)}*$ and a trained rating matrix $r_{predict} \in R^{n\times 3}$

1. reshape $x^{(1)}, ..., x^{(n_m)}, \theta^{(1)}, ..., \theta^{(n_u)}$ as $\mu^{(1)}, ..., \mu^{(n_m)}, \mu^{(n_m+1)}, ..., \mu^{(n_m+n_u)}$
2. **while** $k <$ MaxIter
3. $\mu_k^{(i)} := \mu_k^{(i)} - \alpha \left(\sum_{j:F(i,j)=1} ((\mu^{(j)})^T \mu^{(i)} - y^{(i,j)})\mu_k^{(j)} + \lambda \mu_k^{(i)} \right)$
4. **if** $J(\mu^{(1)}, ..., \mu^{(n_m)}, \mu^{(n_m+1)}, ..., \mu^{(n_m+n_u)})$ converged
5. **break**
6. end **if**
7. end **while**
8. unfold $\mu^{(1)}, ..., \mu^{(n_m)}, \mu^{(n_m+1)}, ..., \mu^{(n_m+n_u)}$ as $x^{(1)}, ..., x^{(n_m)}, \theta^{(1)}, ..., \theta^{(n_u)}$
9. # Get rating prediction $r_{predict} = (\theta^{(j)})^T x^{(i)} \in R^{n\times 3}$

C. RECOMMENDATION RESULT DEPLOYMENT IN MCS

After obtaining the recommendation results for each subsolver, deploying three solvers efficiently in a co-simulation process is the primary focus in this part. Since the numerical algorithm runs under an iteration manner, a complete integration time interval from t_n to t_{n+1} can be analyzed. Solving the co-simulation problem at every round $\Delta t := t_{n+1} - t_n$ involves three steps: (1) Solving (11a), (11b) (11c) using separate ODEs solvers; (2) Estimating the Local Truncation Error (LTE) of a selected sub-solver by comparing half and whole step-size; and (3) Evaluating next step-size t using the bisection strategy.

The following part details this adaptive time-stepping co-simulation algorithm using ODE Sub-solver B within the BDF2 scheme as the selected step-size controller, with three embedded ODEs solvers using three different derivative schemes. It should be noticed that Sub-solver B uses RK4 to

start, since BDF2 is a multi-step method which cannot start on its own.

Algorithm 2 Adaptive Time-Stepping Co-Simulation

Startstopwatch **timer**;

Input: each sub-solver with initial value at time stamp t_0 with initial value; initial LTE threshold $e_{\max}$,$e_{\min}$ and step-size threshold $\Delta t_{\max}$, $\Delta t_{\min}$

Output: dynamical displacement, simulation time, sequence of adaptive time-stepping size.

1. **for**$t_n = t_0{:}\Delta t{:}t_{\max}$ **do**
2. # Subsolver A: RK4

$$k_1 = \Delta t(F_A \mathbf{y}_A(t_n) + F_{AB}\mathbf{y}_B(t_n))$$
$$k_2 = \Delta t\left(F_A\left(\mathbf{y}_A(t_n) + \frac{k_1}{2}\right) + F_{AB}\mathbf{y}_B(t_n)\right)$$
$$k_3 = \Delta t\left(F_A\left(\mathbf{y}_A(t_n) + \frac{k_2}{2}\right) + F_{AB}\mathbf{y}_B(t_n)\right)$$
$$k_4 = \Delta t\left(F_A\left(\mathbf{y}_A(t_n) + \frac{k_2}{2}\right) + F_{AB}\mathbf{y}_B(t_n)\right)$$
$$\mathbf{y}_A(t_{n+1}) = \mathbf{y}_A(t_n) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

3. # Subsolver B: BDF2
$\mathbf{y}_B(t_{n+1}) = \frac{\frac{4}{3}\mathbf{y}_B(t_n) - \frac{1}{3}\mathbf{y}_B(t_{n-1}) + \frac{2}{3}\Delta t(F_{BA}\mathbf{y}_A(t_n) + F_{BC}\mathbf{y}_C(t_{n-1}))}{I - \frac{2}{3}\Delta t F_B}$
4. # Compute $\mathbf{y}_B(t_{n+1/2})$ using BDF2
$\mathbf{y}_B(t_{n+1/2}) = \frac{\frac{4}{3}\mathbf{y}_B(t_n) - \frac{1}{3}\mathbf{y}_B(t_{n-1}) + \frac{2}{3}\frac{\Delta t}{2}(F_{BA}\mathbf{y}_A(t_n) + F_{BC}\mathbf{y}_C(t_{n-1}))}{I - \frac{2}{3}\frac{\Delta t}{2}F_B}$
5. # Estimate LTE using Euclidean norm
$LTE(t_{n+1}) \approx \sum_{i=1}^{n+1} \left\|\mathbf{y}_B(t_{n+1}) - \mathbf{y}_B(t_{n+1/2})\right\|_2$
6. # Control step-size using bisection strategy
if $LTE(t_{n+1}) > e_{\max}$: $\Delta t_{n+1} = \frac{1}{2}\Delta t_n$
else if $LTE(t_{n+1}) < e_{\min}$:$\Delta t_{n+1} = 2\Delta t_n$
end **if**
if $\Delta t_{n+1} > \Delta t_{\max}$: $\Delta t_{n+1} = \Delta t_{max}$
else if $\Delta t_{n+1} < \Delta t_{\min}$: $\Delta t_{n+1} = \Delta t_{\min}$
end **if**
7. # Sub-solver C: Backward Euler
$\mathbf{y}_C(t_{n+1}) = \frac{\mathbf{y}_C(t_n) + \Delta t F_{CB}\mathbf{y}_B(t_n)}{I - \Delta t F_C}$
8. end**for**

End **timer**

Moreover, theRK4 method can also be used at every step in order to achieve step-size controlling. This algorithm is similar to Algorithm 2. The main difference is that this algorithm computes $\mathbf{y}_A(t_{n+1/2})$ rather than $\mathbf{y}_B(t_{n+1/2})$ with a half step-size to estimate LTE. The comparison results of these two step-size controller is shown in Table 4.

D. NUMERICAL ANALYSIS FOR MCS

In this part, the relationship between stability region with step-size and error estimation with step-size is analyzed, and thus it will provide numerical guidance about how to set a proper step-size tolerance and error tolerance for the bisection strategy in Section IV.C.

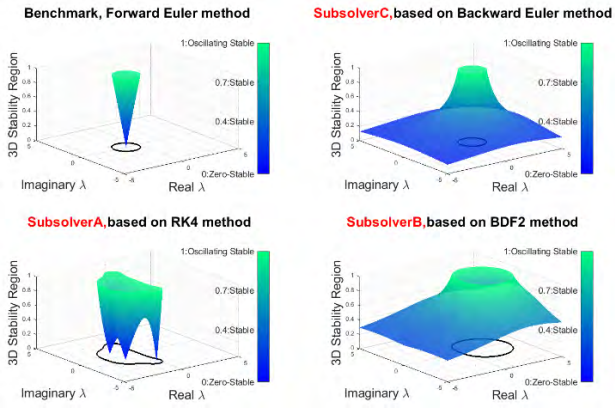

FIGURE 8. Stability region of different ODE solvers.

1) COMPUTATIONAL STABILITY

In order to set a proper step-size tolerance for the bisection strategy, the relationship between stability region and step-size of all the three solvers used in co-simulation needs to be figured out.

First of all, a classical test ODE is used to analyze the stability region of each sub-solver [29]. λ is a given constant in a test ODE. The stability regions of every ODE sub-solver with different derivative scheme are shown in Fig.8. Forward Euler method and RK4 method have round-shape stability region on the complex plane because explicit schemes usually hold conditional stable region. In contrast to explicit schemes, Backward Euler method and BDF2 method are implicit schemes which are almost unconditional stable except in a round region.

Based on the individual stability region consideration mentioned above, analyzing the overlap region is necessary when three different solvers are used. The deep blue in Fig.9 indicates the overlap region, which is the same as the RK4 stability region for the three solvers (i.e. Backward Euler, BDF2, and RK4). This figure shows that the stability region of explicit method will limit the stability performance of the whole co-simulation process.

In order to set a proper threshold for step-size, different step-size t is used to illustrate its region scale change, as shown in Fig. 10. The number "0.4", "0.7" and "1" labeled on the plot demonstrate the strength of stability. Specifically, "1" means oscillation stable, bigger than "1" means unstable, "0.7" is less stable than "0.4", and "0" means unconditional stable (also called zero-stable). The comparison of different step-sizes used for Sub-solver A based on the RK4 method shows that the radius of stability region is proportional to $1/t$. When t increases ten times, the stability region shrinks remarkably towards the original point while when t becomes smaller, the stability region expands rapidly in the complex plane.

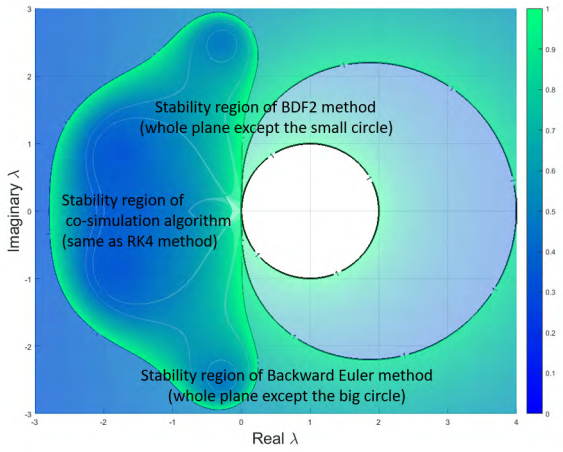

FIGURE 9. Stability region overlap of three ODE solvers.

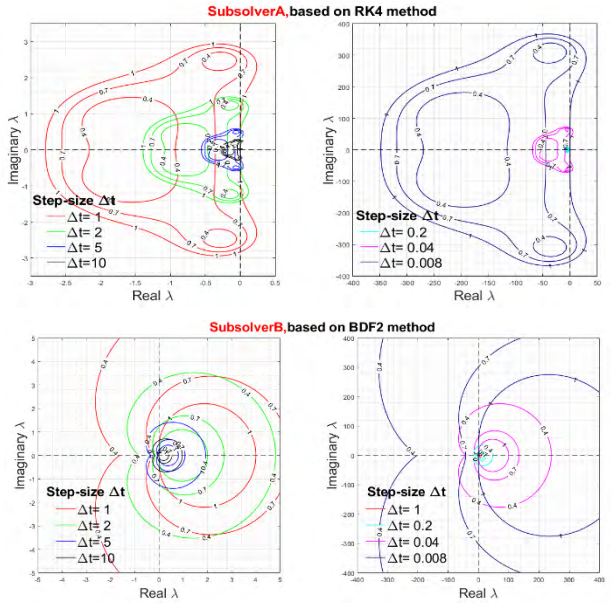

FIGURE 10. Stability region comparison of different step-size.

However, the relationship between step-size and the stability region is different in Sub-solver B which uses an imbedded BDF2 method. Since BDF2 is an implicit method, its stability region spans almost the whole complex plane except a circle. When t increases ten times, the circle region shrinks remarkably towards the original point, meaning the stability region is actually increasing. Thus, the increases of the stability region is proportional to the increases of t. Although RK4 and BDF2 show different relationship between step-size and the stability region, the stability region of RK4 is always a real sub-region of BDF2's at the same step-size t. Based on this observation, controlling the step-size of either RK4 or BDF2 can achieve the same stable region.

2) TOLERANCE PROPORTIONALITY

In this part, the relationship between error and step-size is analyzed in detail so as to set proper maximum and minimum thresholds of error for the bisection strategy. Usually round-off error and truncation error compose the inaccuracy problem when using numerical algorithms to solve differential equations. Hence, the relationship between derivative scheme error and step-size is discussed here. Equation (16) gives the error measurement of each derivative scheme.

$$e(\Delta t) \approx \sqrt{\frac{\sum_{t=1}^{t=T} (y(t+\Delta t) - y(t+\Delta t/2))^2}{(T/\Delta t)}} \tag{16}$$

In this equation,$y(t+t)$ and $y(t+t/2)$ are iterative time-stepping results using full step-size and half step-size at time t, respectively. The advantage of this error measurement is that it enables ignoring the length of time interval and focusing on the error change with the different value of step-size t. The 1st order derivative scheme, such as Forward Euler or Backward Euler, and 2nd order derivative scheme, such as BDF2 or trapezoidal method are conducted with different refinements of t.

The 1st order derivative scheme has the general form as follows:

$$\dot{y} \approx \frac{y(t+\Delta t) - y(t)}{t} + O(\Delta t) \tag{17}$$

The 2nd order derivative scheme has the general form as follows:

$$\dot{y} \approx \frac{y(t+\Delta t) - y(t-\Delta t)}{2\Delta t} + O(\Delta t^2) \tag{18}$$

The simulation results are shown in Fig.11 using a loglog plot to illustrate the relationship between step-size and error type in terms of different derivative orders. The check-shape curve of the 1st order derivative scheme shows that if $t<10^{-8}$, round-off error has a dominant impact of the scheme accuracy; if $\Delta t>10^{-8}$, truncation error has a dominant impact; and $\Delta t=10^{-8}$ is an optimal trading off point. The check-shape curve of the 2nd order derivative scheme shows a similar trend. Since the minimum threshold of error is usually larger than 10^{-8}, truncation error then becomes the main component – this means that the LTE (Local Truncation Error) estimation presented at Step 5 of Algorithm 2 in Section IV.C is reasonable.

The logarithm of error and the logarithm of step-size has a linear correlation, as shown in Fig.11. The linear curve can be represented as Equation (19),

$$\log(e(\Delta t)) = a + b \log \Delta t \tag{19}$$

then using exponential on both sides of the equation, Equation (20) can be obtained.

$$e(\Delta t) = \exp(a) \exp(b \log \Delta t) = \exp(a) \Delta t^b \tag{20}$$

In Equation (20), a is the vertical intercept and b is the slope of curve. From Fig.11, a conclusion can be drawn that

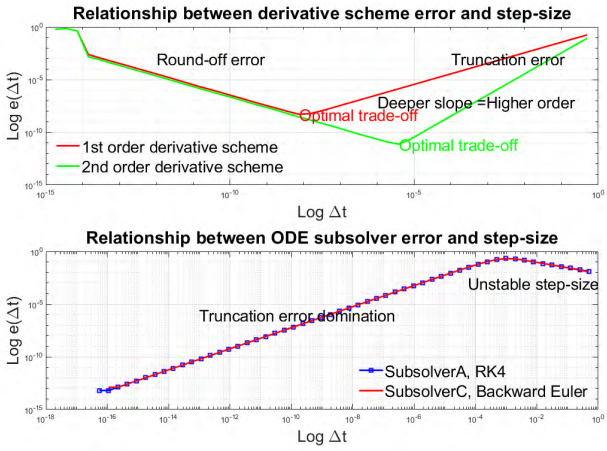

FIGURE 11. Comparison of round-off error and truncation error for different solvers.

the higher order of the derivative scheme, the deeper of the slope. For higher order schemes, error changes faster with the change of step-size.

E. IMPLEMENTATION AND EVALUATION

To evaluate the proposed adaptive time-stepping co-simulation approach, a number of numerical experiments are conducted to compare different simulation patterns and ODE solvers using the reformation problem introduced in Section III.A. Values of parameters and initial values of variables used in the experiments are shown in Table 3.

TABLE 3. System parameters and values for model problem.

Parameter	Physical Description	Value	Unit
m_1, m_2, m_3	Mass	0.1, 0.2, 0.3	kg
k_1, k_2, k_3	Spring stiffness	1.0, 2.0, 3.0	N/m
d_1, d_2	Damping coefficient	0.1, 0.5	N/s
$u(t_0), v(t_0), w(t_0)$	Initial displacement	1.0, 0.0, 0.0	m
$\dot{u}(t_0), \dot{v}(t_0), \dot{w}(t_0)$	Initial velocity	0.0, 0.0, 0.0	m

To ensure the validity of comparison, the same parameter value, initial variable value, initial step-size, same error threshold and step-size threshold are used in the simulation experiments conducted in this section. First, ODE solvers RK4, BDF2, Backward Euler (BE) are used separately to solve Equation (9) as a monolithic simulation problem, using a fixed step-size. Then, three ODE solvers are used together to Jacobi method is used as the parallel communication pattern in the fixed step-size part. From the results in Table 3, the simulation effects of these two methods are quite similar and hence only the Jacobi pattern is picked up to do the adaptive step-size experiments. The comparison of results for all the experiments mentioned above are also listed in Table 3.

From the experiment results of simulation time and steps number shown in Table 4, the proposed adaptive time- stepping co-simulation Algorithm 2 has obvious advantages in step usage compared to other simulation manners when the initial step-size is smaller than 0.05. In addition, the step usage in Algorithm 2 maintains a stable pattern when the initial step-size changes 20 times from 0.005 to 0.1, which means the step usage does not depend on the initial step-size choice in this algorithm. It would be a great advantage when this algorithm is embedded in large-scale co-simulation problems for which step-size estimation trials are necessary.

In terms of accuracy and simulation speed, the proposed algorithm almost maintains a constant time-spending manner using different initial step-sizes, as shown in Table 4. Hence, when high-fidelity simulation tasks are required, the proposed algorithm will demonstrate its efficiency regarding computational cost. In order to evaluate the simulation accuracy, three variable curves of displacement are shown in Fig.12. Specifically, Fig. 12(a) shows the results obtained from the proposed algorithm using RK4 as the step-size controller and Fig. 12(b) shows the results obtained from the proposed algorithm using BDF2 as the step-size controller. From the curves demonstration, the accuracy of the 4th order can be kept using RK4 as the step-size controller because the LTE estimation is based on the 4th order derivative scheme. The curves using BDF2 as the controller are less smooth compared with that using RK4, because the LTE estimation is based on the 2nd order derivative scheme. The main advantage of the BDF2 controller is that its iteration steps are half of RK4's and its simulation time is only two third of RK4's. As such, the choices of solver schemes and controller strategies involve making a trade-off between accuracy and efficiency both of which need to be considered to achieve overall good simulation performance.

V. ADDITIONAL EXPERIMENTS WITH DETEST

In this part, the proposed approach is implemented and tested on an ODE solver library named DETEST [47] which has been briefly introduced in Section III.B. It specifically consists of 6 well-known ODE solvers and 25 test problems. The test problems used in DETEST, which are presented in Appendix, are divided into the following five categories: A: single equations; B: small systems; C: moderate systems; D: orbit equations; and E: higher order equations. Each test problem is associated with 3 different tolerances, namely 10^{-3}, 10^{-6}, 10^{-9}.Thus there are a total of 75 test cases that can be solved by a set of solvers including BULIRSCH-STOER, ADAMS: KROGH, ADAMS:GEAR, RK4:KUTTA, RK6:BUTCHER and RK8:SHANKS. A snapshot of DETSET is shown in Table 5.

To evaluate the proposed recommendation approach, both data sparsity and algorithm learning parameters are considered. The Evaluation Metrics for the CF multi-solver recommendation algorithm includes the Relative Mean Absolute Error (RMAE) with a formula shown in Equation (21) and the Relative Mean Square Error (RMSE) with a formula shown

TABLE 4. Comparison of step usage for different simulation patterns and ode solvers.

	Monolithic simulation			Co-simulation			
Step-size type	Fixed			Fixed		**Adaptive**	
Communication pattern	N/A	N/A	N/A	Guass-Seidel	Jacobi	Jacobi Error threshold [0.001, 0.01] Step-size threshold [1e-5, 0.5]	
ODE solvers	Only RK4	Only BDF2	Only Backward Euler(BE)	A:RK4 B:BDF2 C:BE	A:RK4 B:BDF2 C:BE	A:RK4(step-size controllor) B:BDF2 C: BE	A:RK4 B:BDF2(step-size controllor) C: BE
Initial step-size	Simulation time /computer clock (seconds) & Steps						
0.005	0.042346	0.049395	0.041834	0.10318	0.098599	0.065083	0.049759
	Steps: 2000					**Steps:290**	**Steps:107**
0.01	0.020266	0.022242	0.019217	0.051672	0.050798	0.063148	0.047613
	Steps: 1000					**Steps:289**	**Steps:106**
0.05	0.007283	0.005678	0.004412	0.020645	0.018816	0.077377	0.046665
	Steps: 200					**Steps:273**	**Steps:112**
0.1	0.006205	0.004090	0.0031683	0.016051	0.015824	0.061594	0.046746
	Steps: 100					**Steps:260**	**Steps:104**

TABLE 5. A snapshot of detest.

ClassA BULIRSCH-STOER					**ClassA ADAMS: KROGH**					**ClassA ADAMS:GEAR**				
10^{-3}	FCN	NO.	PER	MAX	10^{-3}	FCN	NO.	PER	MAX	10^{-3}	FCN	NO.	PER	MAX
A1	150	6	0	0.2	A1	63	36	0	0.3	A1	114	46	0	0.9
A2	126	6	0	0	A2	71	41	0	0	A2	71	33	0	1
A3	319	7	14.3	1.2	A3	192	99	2	1.1	A3	315	106	3.8	1.5
A4	126	6	0	0	A4	27	25	0	0.1	A4	62	23	17.4	1.3
A5	126	6	0	0	A5	58	34	0	0	A5	62	30	0	1

From Class A/tolerance10^{-3} to Class E/ tolerance10^{-9}

ClassE RK4:KUTTA					**ClassE RK6:BUTCHER**					**ClassE RK8:SHANKS**				
10^{-9}	FCN	NO.	PER	MAX	10^{-9}	FCN	NO.	PER	MAX	10^{-9}	FCN	NO.	PER	MAX
E1	4543	413	0	0.9	E1	1620	81	0	0.7	E1	1015	29	0	0.6
E2	15512	1402	0.2	1.3	E2	5902	259	0.8	1	E2	3927	85	5.9	10.2
E3	11007	997	0	1	E3	3416	148	1.4	1.1	E3	2019	47	0	0.9
E4	494	44	0	0.8	E4	239	11	0	0.7	E4	243	5	0	0.4
E5	737	67	0	0.8	E5	391	11	0	0.5	E5	277	5	0	0.2

in Equation (22).

$$RMAE = \frac{1}{\text{STD}} \sum_{(i,j):F(i,j)=1} \left| (\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right| \tag{21}$$

$$RMSE = \frac{1}{VAR} \sum_{(i,j):F(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 \tag{22}$$

In the above two equations, STD means the standard deviation while VAR is the variance of dataset.

In order to test the influence of data sparsity on recommendation accuracy, a three-component filter matrix $F \in R^{n\times 3}$ is built with nearly 1/3, 1/2 and 2/3 sparsity. The random 0-1 filter follows standard normal distribution as shown in Fig.13.

Some conclusions can be drawn based on the various experiment results. There are three primary factors influencing the proposed CF recommendation algorithm performance, namely the regression model feature number, the regularization parameter lambda, and the advanced optimization method maximum iteration. The ceteris paribus conditions are used in this experiment - in simple words, one aspect is tested at a time with other aspects being equal. The relationships between factors and data sparsity are shown in Fig.14. The curves shows similar trends with different sparsity. Among Fig.14(a) (b) and (c), the RMAE reaches the lowest all under sparsity of 34%, although the parameter settings are quite different.

Table 6 shows a comparison of RMAE and RMSE with respect to different parameter settings based on the evaluation conducted using computational experiments. The bolded parameter is the best result with the lowest error estimation. From the results shown in Table 6, a conclusion can be drawn that RMAE reaches the smallest compared with other parameter sets in this table when parameter settings are Sparsity =34.00%, Feature =6, MaxIter = 80, and Lambda =0.5.

Using the comparatively optimal parameter setting in Table 6, we also evaluated the influence by using different optimization methods, which shows in Table 7. From this table, we can easily find that the comparison differences are relatively minor than the differences due to sparsity.

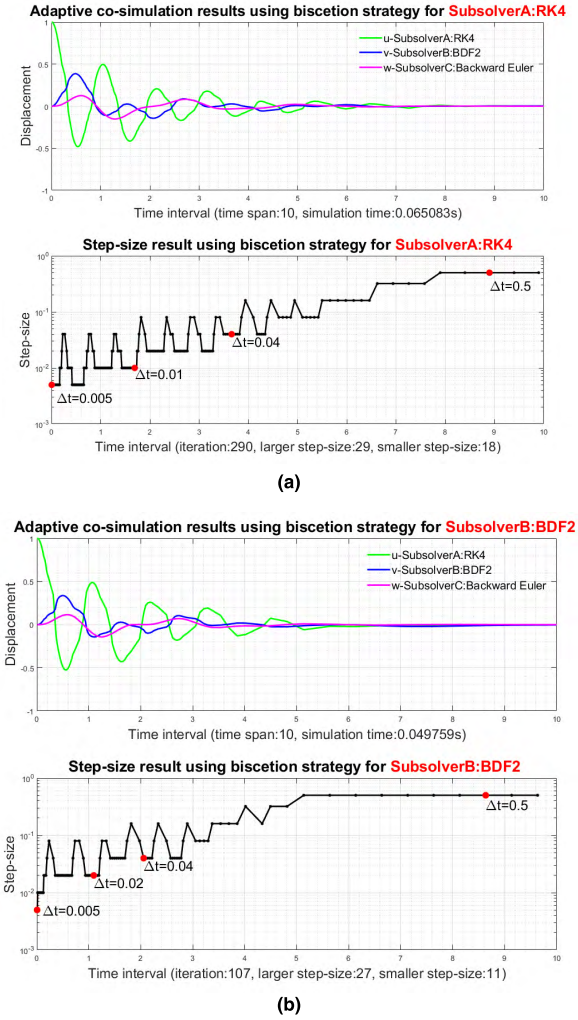

FIGURE 12. Adaptive step-size simulation results. (a) Simulation results using RK4 as step-size controller. (b) Using BDF2 as step-size controller.

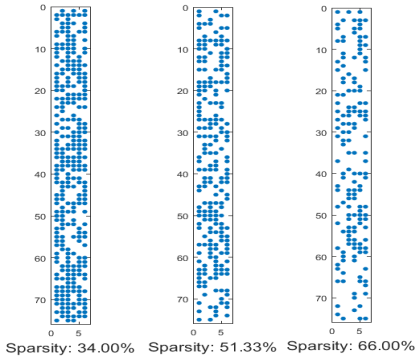

FIGURE 13. Three filter matrix with nearly 1/3, 1/2, 2/3 sparsity.

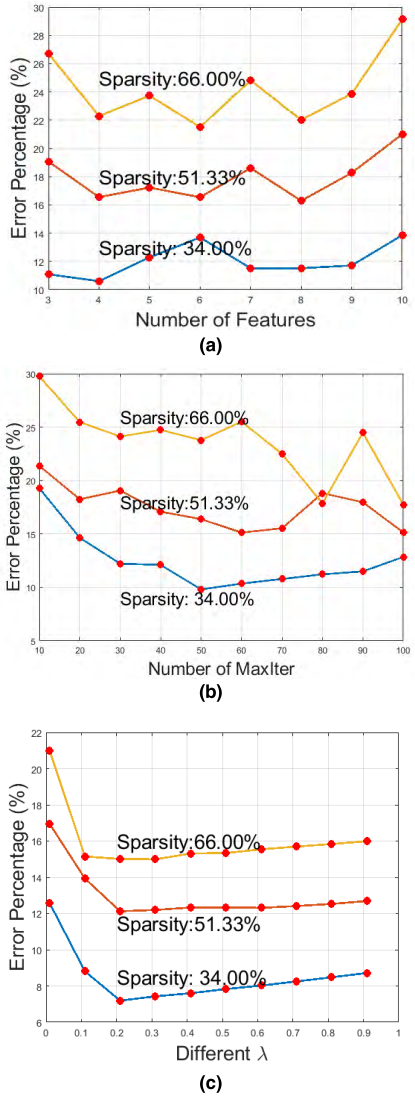

FIGURE 14. Factor and data sparsity comparison results.
(a) FeatureNumber-RMAE Results, with Lambda = 0.01; Maxiter = 50.
(b) Maxiter-RMAE Results, with Lambda = 0.01; Feature Number = 5.
(c) Lambda-RMAE Results, with Maxiter = 50; Feature Number = 5.

TABLE 6. Error comparison for testset and allset.

Sparsity	Feature	Max-Iter	Lambda	TestSet RMAE	AllSet RMAE	AllSet RMSE
66.00%	4	50	0.1	0.3328	0.1552	0.3957
	6	**80**	**0.1**	0.3215	**0.1487**	0.3889
	8	100	0.5	**0.3176**	0.1535	**0.3868**
51.33%	4	20	0.1	0.3616	0.1946	0.6283
	6	40	0.1	0.2857	0.1402	0.3657
	8	**60**	**0.2**	**0.2326**	**0.1108**	**0.2829**
34.00%	**4**	**50**	**0.2**	0.1744	0.0891	0.2251
	6	80	0.5	0.1549	**0.0762**	**0.1992**
	8	100	0.9	**0.1543**	0.0869	0.2055

At the end of this part, the parameter set (Sparsity =34.00%, Feature =6, MaxIter =80, Lambda =0.5) is used to train the regression-based CF model to get the rating results for all solvers paired with different test problems. The resultant rating score is shown in Table 8. From the result, it can be concluded that BULIRSCH-STOER is better at dealing with small scale problems (Class A) than higher order problems (Class E), while some other methods, like ADAMS: GEAR and RK8: SHANKS, do not

TABLE 7. Allset rmse comparison for different optimization method (sparsity: 34.00%, feature 6, maxiter 80, lambda 0.5).

Iteration	steepest descend	stochastic gradient descend	sequence quadratic program
10	0.2729	0.2955	0.2131
20	0.2705	0.2244	0.2094
30	0.2433	0.2054	0.1613
40	0.1353	0.1774	0.1581
50	0.1216	0.1725	0.1355
60	0.1116	0.1292	0.1117
70	0.0970	0.0539	0.1003
80	0.0687	0.0515	0.0845
90	0.0576	0.0141	0.0803
100	0.0573	0.0109	0.0236

TABLE 8. Rating result using proposed approach (sparsity: 34.00%, feature 6, maxiter 80, lambda 0.5).

ODE solver	CLASS A	CLASS B	CLASS C	CLASS D	CLASS E
BULIRSCH-STOER	3.0268	3.1222	2.4382	3.2181	2.7653
ADAMS: KROGH	2.0171	2.0077	2.0161	2.0045	2.0460
ADAMS: GEAR	1.8876	1.9673	1.9624	1.9729	1.9346
RK4: KUTTA	2.0941	1.9930	2.0517	2.0902	2.2574
RK6: BUTCHER	2.2692	2.1309	1.7887	2.2008	2.4507
RK8: SHANKS	2.7621	2.3583	2.3375	2.2964	2.6730

have significant performance difference dealing with different classes. This means that they are more viable and stable when co-simulation problems contain both stiff and non-stiff parts.

VI. CONCLUSION

In this paper, a regression-based CF approach for recommending solvers and time-stepping communication mechanism is proposed to provide a general framework for portioned engineering system simulation using MCS. Specifically, by reformulating a monolithic problem into partitioned system theoretically, a regression-based CF algorithm is used to recommend suitable solvers for subsystems based their historical usage data and engineering characteristics. Then, an adaptive time-stepping co-simulation algorithm embedded with three ODE sub-solvers for solving this coupling scheme is proposed based on the Jacobi communication pattern. The time-stepping algorithm uses LTE estimation based on a selected sub-solver and implements run-time control of step-size based on a bisection strategy. Moreover, through identifying the relationship between the stability region and step-size, numerical guidance is provided to set a proper step-size threshold and error threshold for the bisection strategy. Lastly, the comparison of results obtained from computational experiments using DETEST shows that the regression-based CF recommendation algorithm achieves good efficacy and effectiveness in dealing with multi-solver co-simulation problems.

In summary, the primary advantages achieved by the approach proposed in this paper are:

(1) It provides a feasible communication solution for co-simulation coupling problem. Numerical analysis proves that embedding three different scheme ODE solvers in a co-simulation problem is workable. The proposed Jacobi-based step-controlling scheme is efficient to deal with co-simulation problems, especially for the complex engineering systems without a monolithic model expression.

(2) The regression-based CF recommendation algorithm is effective and helpful for automatically selecting solvers during simulation run-time, which collects and utilizes historical simulation data in a novel way.

(3) The time-stepping algorithm can deal with iteration efficiently. The number of iteration or steps does not depend on the initial step-size choice. It would be a great advantage when this algorithm is embedded in large-scale and high-fidelity co-simulation problems with the need of step-size estimation trials.

It enables more flexible selection of solvers as step-size controllers according to specific requirements in terms of accuracy and efficiency. While moving a step forward towards more a general co-simulation framework, the research work reported in this paper also has some limitations. First, accuracy of the trained regression-based CF algorithm can still be improved with more information about ODE libraries. Second, the reformed mathematical description can only be used in linear ODEs, or the ODEs semi-discretized from linear PDEs. Third, higher computational cost higher than fixed-step algorithm may incur for the applications in which only low-fidelity simulation is required as it uses LTE estimation and the bisection strategy at each time step.

APPENDIX

The test problems used in DETEST are presented as follows. The analytic solutions are given if they are available.

PROBLEM CLASS A: SINGLE EQUATIONS

A1: The negative exponential problem

$\dot{y} = -y,\ y(0) = 1$ (solution: $y = Ce^{-x}, C = 1$)

A2: A special case of the Riccati equation

$\dot{y} = -y^3/2,\ y(0) = 1$ (solution: $y = 1/\sqrt{x + C}, C = 1$)

A3: An oscillatory problem

$\dot{y} = ycosx,\ y(0) = 1$ (solution: $y = Ce^{sinx}, C = 1$)

A4: A logistic curve

$$\dot{y} = \frac{y}{4}\left(1 - \frac{y}{20}\right), \quad y(0) = 1$$

(solution: $y = \frac{20}{1+19Ce^{-x/4}}, C = 1$)

A5: A spiral curve

$$\dot{y} = \frac{y - x}{y + x}, \quad y(0) = 4$$

(solution in polar coordinates: $r = Ce^{-\theta}, C = 4e^{\pi/2}$)

PROBLEM CLASS B: SMALL SYSTEMS

B1: The growth of two conflicting populations

$$\dot{y}_1 = 2(y_1 - y_1y_2), \quad y_1(0) = 1$$
$$\dot{y}_2 = -(y_2 - y_1y_2), \quad y_2(0) = 3$$

B2: A linear chemical reaction

$$\dot{y}_1 = -y_1 + y_2, \quad y_1(0) = 2$$
$$\dot{y}_2 = y_1 - 2y_2 + y_3, \quad y_2(0) = 0$$
$$\dot{y}_3 = y_2 - y_3, \quad y_3(0) = 1$$

B3: A linear chemical reaction

$$\dot{y}_1 = -y_1, \quad y_1(0) = 1$$
$$\dot{y}_2 = y_1 - y_2^2, \quad y_2(0) = 0$$
$$\dot{y}_3 = y_2^2, \quad y_3(0) = 0$$

B4: The integral surface of a torus

$$\dot{y}_1 = -y_2 - y_1 y_3 / \sqrt{y_1^2 + y_2^2}, \quad y_1(0) = 3$$
$$\dot{y}_2 = y_1 - y_2 y_3 / \sqrt{y_1^2 + y_2^2}, \quad y_2(0) = 0$$
$$\dot{y}_3 = y_1 / \sqrt{y_1^2 + y_2^2}, \quad y_3(0) = 0$$

B5: Euler equations of motion for a rigid body without external force

$$\dot{y}_1 = y_2 y_3, \quad y_1(0) = 0$$
$$\dot{y}_2 = -y_1 y_3, \quad y_2(0) = 1$$
$$\dot{y}_3 = -0.51 y_1 y_2, \quad y_3(0) = 1$$

PROBLEM CLASS C: MODERATE SYSTEMS

C1: A radioactive decay chain

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \vdots \\ \dot{y}_{10} \end{bmatrix} = \begin{bmatrix} -1 & 0 & \cdots & 0 & 0 \\ 1 & -1 & \cdots & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & & -1 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{10} \end{bmatrix},$$

$$\mathrm{y}(0) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

C2: Another radioactive decay chain

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \vdots \\ \dot{y}_{10} \end{bmatrix} = \begin{bmatrix} -1 & 0 & \cdots & 0 & 0 \\ 1 & -2 & & 0 & 0 \\ & & 2 \quad -3 & & \\ \vdots & & 0 \quad \ddots & \vdots & \\ 0 & 0 & & -9 & 0 \\ 0 & 0 & \cdots & 9 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{10} \end{bmatrix},$$

$$\mathrm{y}(0) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

C3: Derived from a parabolic partial differential equation

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \vdots \\ \vdots \\ \dot{y}_{10} \end{bmatrix} = \begin{bmatrix} -2 & 1 & & 0 & 0 \\ 1 & -2 & \cdots & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & & -2 & 1 \\ 0 & 0 & \cdots & 1 & -2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{10} \end{bmatrix},$$

$$\mathrm{y}(0) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \tag{23}$$

C4: As in C3 except with 51 equations.

C5: Five body problem: the motion of 5 outer planets about the sun.

The 3 spatial coordinates of the j th body are

$$\ddot{y}_{ij} = k_2 \left(-\left(m_o + m_j\right) \frac{y_{ij}}{r_j^3} + \sum_{\substack{k=1 \\ k \neq j}}^{5} m_k \left[\frac{y_{ik} - y_{ij}}{d_{jk}^3} - \frac{y_{ik}}{r_k^3} \right] \right)$$

where $r_j^2 = \sum_{i=1}^{3} y_{ij}^2$ and $d_{kj}^2 = \sum_{i=1}^{3} (y_{ik} - y_{ij})^2$, $k, j = 1, \ldots, 5$

When this system is rewritten using only first order differential equations the dependent vector has 30 components.

$k_2 = 2.95912208286$ (gravitational constant),

$m_0 = 1.00000597682$ (mass of the sun and the 4 inner planets),

$m_1 = .000954786104043$ (Jupiter),

$m_2 = .000285583733151$ (Saturn),

$m_3 = .0000437273164546$ (Uranus),

$m_4 = .0000517759138449$ (Neptune),

$m_5 = .00000277777777778$ (Pluto).

The initial values are:

$y_{11} = 3.42947415189$, $\dot{y}_{11} = -.557160570446$,
$y_{21} = 3.35386959711$, $\dot{y}_{21} = .505696783289$,
$y_{31} = 1.35494901715$, $\dot{y}_{31} = .230578543901$,
$y_{12} = 6.64145542550$, $\dot{y}_{12} = -.415570776342$,
$y_{22} = 5.97156957878$, $\dot{y}_{22} = .365682722812$,
$y_{32} = 2.18231499728$, $\dot{y}_{32} = .169143213293$,
$y_{13} = 11.2630437207$, $\dot{y}_{13} = -.325325669158$,
$y_{23} = 14.6952576794$, $\dot{y}_{23} = .189706021964$,
$y_{33} = 6.27960525067$, $\dot{y}_{33} = .0877265322780$,
$y_{14} = -30.1552268759$, $\dot{y}_{14} = -.0240476254170$,
$y_{24} = 165699966404$, $\dot{y}_{24} = -.287659532608$,
$y_{34} = 1.43785752721$, $\dot{y}_{34} = -.117219543175$,
$y_{15} = -21.1238353380$, $\dot{y}_{15} = -.176860753121$,
$y_{25} = 28.4465098142$, $\dot{y}_{25} = -216393453025$,
$y_{35} = 15.3882659679$, $\dot{y}_{35} = -.0148647893090$.

PROBLEM CLASS D: ORBIT EQUATIONS

D1:

$$\dot{y}_1 = y_3, \quad y_1(0) = 1-\varepsilon$$
$$\dot{y}_2 = y_4, \quad y_2(0) = 0$$
$$\dot{y}_3 = -y_1/(y_1^2+y_2^2)^{3/2}, \quad y_3(0) = 0$$
$$\dot{y}_4 = -y_2/(y_1^2+y_2^2)^{3/2}, \quad y_4(0) = \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}$$
$$\varepsilon = 0.1(\varepsilon \text{istheeccentricityoftheorbit})$$

D2: As above with $\varepsilon = 0.3$
D3: As above with $\varepsilon = 0.5$
D4: As above with $\varepsilon = 0.7$
D5: As above with $\varepsilon = 0.9$
(All are derived from the orbit equations

$$\ddot{x} = -\frac{x}{r^3}, \quad x(0) = 1-\varepsilon, \quad \dot{x}(0) = 0$$
$$\ddot{y} = -\frac{y}{r^3}, \quad x(0) = 0, \quad \dot{y}(0) = \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}$$
$$r^2 = x^2+y^2$$

with solution $x = cosu - \varepsilon, \dot{x} = \frac{-sinu}{1-\varepsilon cosu}$

$$y = \sqrt{1-\varepsilon^2}sinu, \dot{y} = \frac{\sqrt{1-\varepsilon^2}cosu}{1-\varepsilon cosu}$$

where $-\varepsilon sinu - t = 0$)

PROBLEM CLASS E: HIGHER ORDER EQUATIONS

E1: $\dot{y}_1 = y_2$

$$\dot{y}_2 = -\left(\frac{y_2}{x+1} + \left(1 - \frac{0.25}{(x+1)^2}\right)y_1\right)$$
$$y_1(0) = J_{\frac{1}{2}}(1) = 0.6713967071418030$$
$$y_2(0) = \dot{J}_{\frac{1}{2}}(1) = 0.09540051444747446$$

(derived from Bessel's equation of order 1/2 with the origin shifted one unit to the left):

$$(x+1)^2\ddot{y} + (x+1)\dot{y} + \left((x+1)^2 - 0.25\right)y = 0$$

E2: $\dot{y}_1 = y_2, y_1(0) = 2$

$$\dot{y}_2 = (1-y_1^2)y_2 - y_1, \quad y_2(0) = 2$$

(derived from Van der Pol's equation

$$\ddot{y} - \left(1-y^2\right)\dot{y} + y = 0)$$

E3: $\dot{y}_1 = y_2, y_1(0) = 0$

$$\dot{y}_2 = \frac{y_1^3}{6} - y_1 + 2\sin?(2.78535x), \quad y_2(0) = 0$$

(derived from Duffing's equation

$$\ddot{y} + y - \frac{y^3}{6} = 2\sin(2.78535x))$$

E4: $\dot{y}_1 = y_2, y_1(0) = 30$

$$\dot{y}_2 = 0.032 - 0.4y_2^2, \quad y_2(0) = 0$$

(derived from the falling body equation $\ddot{y} = 0.032 - 0.4\dot{y}^2$)

E5: $\dot{y}_1 = y_2, y_1(0) = 0$

$$\dot{y}_2 = \sqrt{1+y_2^2}/(25-x), \quad y_2(0) = 0$$

(derived from a linear pursuit equation

$$1 + \dot{y}^2 = (25-x)^2\ddot{y}^2)$$

REFERENCES

[1] H. Zhang, H. Wang, D. Chen, and G. Zacharewicz, "A model-driven approach to multidisciplinary collaborative simulation for virtual product development," *Adv. Eng. Informat.*, vol. 24, no. 2, pp. 167–179, 2010.
[2] F. González, M. A. Naya, A. Luaces, and M. González, "On the effect of multirate co-simulation techniques in the efficiency and accuracy of multibody system dynamics," *Multibody Syst. Dyn.*, vol. 25, no. 4, pp. 461–483, 2011.
[3] A. Zhu, M. Jarrett, Y. Xu, B. Kochunas, E. Larsen, and T. Downar, "An optimally diffusive Coarse Mesh Finite Difference method to accelerate neutron transport calculations," *Ann. Nucl. Energy*, vol. 95, pp. 116–124, Sep. 2016.
[4] M. B. Van Gijzen, G. L. G. Sleijpen, and J.-P. M. Zemke, "Flexible and multi-shift induced dimension reduction algorithms for solving large sparse linear systems," *Numer. Linear Algebra Appl.*, vol. 22, no. 1, pp. 1–25, 2015.
[5] H. Jafarkarimi, A. H. S. Tze, and R. Saadatdoost, "A naive recommendation model for large databases," *J. Inf. Educ. Technol.*, vol. 2, no. 3, p. 216, 2012.
[6] L. Terveen and W. Hill, "Beyond recommender systems: Helping people help each other," in *HCI in the New Millennium 1*, no. 2001. Reading, MA, USA: Addison-Wesley, 2001, pp. 487–509.
[7] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. Artif. Intell.*, vol. 2009, Aug. 2009, Art. no. 421425. doi: 10.1155/2009/421425.
[8] M.-C. Hsu *et al.*, "Dynamic and fluid–structure interaction simulations of bioprosthetic heart valves using parametric design with T-splines and Fung-type material models," *Comput. Mech.*, vol. 55, no. 6, pp. 1211–1225, 2015.
[9] M. Busch and B. Schweizer, "Coupled simulation of multibody and finite element systems: An efficient and robust semi-implicit coupling approach," *Arch. Appl. Mech.*, vol. 82, no. 6, pp. 723–741, 2012.
[10] H. Zhang, H. Wang, and D. Chen, "Integrating Web services technology to HLA-based multidisciplinary collaborative simulation system for complex product development," in *Proc. 12th Int. Conf. Comput. Supported Cooperat. Work Design*, Apr. 2008, pp. 420–426.
[11] M. Arnold, C. Clauss, and T. Schierz, "Error analysis and error estimates for co-simulation in FMI for model exchange and Co-simulation V2.0," *Arch. Mech. Eng.*, vol. 60, no. 1, pp. 75–94, 2013.
[12] C. M. Macal and M. J. North, "Tutorial on agent-based modelling and simulation," in *Proc. Winter Simulation Conf.*, Orlando, FL, USA, 2005, p. 14. doi: 10.1109/WSC.2005.1574234. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1574234&isnumber=33294
[13] B. Peherstorfer, S. Zimmer, C. Zenger, and H.-J. Bungartz, "A multigrid method for adaptive sparse grids," *SIAM J. Sci. Comput.*, vol. 37, no. 5, pp. S51–S70, 2015.
[14] A. Imren and D. C. Haworth, "On the merits of extrapolation-based stiff ODE solvers for combustion CFD," *Combustin Flame*, vol. 174, pp. 1–15, Dec. 2016.
[15] V. Casulli, "A conservative semi-implicit method for coupled surface–subsurface flows in regional scale," *Int. J. Numer. Methods Fluids*, vol. 79, no. 4, pp. 199–214, 2015.
[16] M. Usui, H. Niki, and T. Kohno, "Adaptive gauss-seidel method for linear systems," *Int. J. Comput. Math.*, vol. 51, nos. 1–2, pp. 119–125, 1994.
[17] I. Slapničar, "Accurate symmetric eigenreduction by a Jacobi method," in *Linear Algebra for Large Scale and Real-Time Applications*. Dordrecht, The Netherlands: Springer, 1993, pp. 417–418.
[18] H. Wang, H. Mao, and H. Zhang, "A variable-step interaction algorithm for multidisciplinary collaborative simulation," *Integr. Comput.-Aided Eng.*, vol. 21, no. 3, pp. 263–279, 2014.
[19] S. E. Minkoff and N. M. Kridler, "A comparison of adaptive time stepping methods for coupled flow and deformation modeling," *Appl. Math. Model.*, vol. 30, no. 9, pp. 993–1009, 2006.

[20] C. Farhat and M. Lesoinne, "Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems," *Comput. Methods Appl. Mech. Eng.*, vol. 182, pp. 499–515, Feb. 2000.
[21] F. Armero and J. C. Simo, "A new unconditionally stable fractional step method for non-linear coupled thermomechanical problems," *Int. J. Numer. Methods Eng.*, vol. 35, no. 4, pp. 737–766, 1992.
[22] J. Wang, Z.-D. Ma, and G. M. Hulbert, "A gluing algorithm for distributed simulation of multibody systems," *Nonlinear Dyn.*, vol. 34, pp. 159–188, Oct. 2003.
[23] A. Pope, *The CORBA Reference Guide: Understanding the Common Object Request Broker Architecture*. Reading, MA, USA: Addison-Wesley, 1998.
[24] J. Hu and H. Zhang, "Ontology based collaborative simulation framework using HLA and Web services," in *Proc. WRI World Congr. Comput. Sci. Inf. Eng.*, vol. 5, Mar./Apr. 2009, pp. 702–706.
[25] L. Zhang *et al.*, "Cloud manufacturing: A new manufacturing paradigm," *Enterprise Inf. Syst.*, vol. 8, no. 2, pp. 167–187, 2014.
[26] H. Wang, H. Zhang, and A. L. Johnson, "A service-oriented approach for the collaborative simulation of complex engineering systems," in *Proc. Congr. Services-I*, Jul. 2009, pp. 78–84.
[27] L. Monostori, J. Váncza, and S. R. T. Kumara, "Agent-based systems for manufacturing," *CIRP Ann. Technol.*, vol. 55, no. 2, pp. 697–720, 2006.
[28] B. Schweizer and D. Lu, "Semi-implicit co-simulation approach for solver coupling," *Arch. Appl. Mech.*, vol. 84, no. 12, pp. 1739–1769, 2014.
[29] S. A. Sicklinger, "Stabilized co-simulation of coupled problems including fields and signals," Ph.D. dissertation, Techn. Univ. München, Munich, Germany, 2014.
[30] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.
[31] A. M. Rashid, S. K. Lam, G. Karypis, and J. Riedl, "ClustKNN: A highly scalable hybrid model-& memory-based CF algorithm," in *Proc. KDD*, 2006, pp. 1–10.
[32] S. Vucetic and Z. Obradovic, "Collaborative filtering using a regression-based approach," *Knowl. Inf. Syst.*, vol. 7, no. 1, pp. 1–22, 2005.
[33] X. Su and T. M. Khoshgoftaar, "Collaborative filtering for multi-class data using belief nets algorithms," in *Proc. 18th IEEE Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2006, pp. 497–504.
[34] P. Symeonidis, A. Nanopoulos, A. Papadopoulos, and Y. Manolopoulos, "Nearest-biclusters collaborative filtering with constant values," in *Proc. Int. Workshop Knowl. Discovery Web*. Berlin, Germany: Springer, 2006, pp. 36–55.
[35] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 89–115, Jan. 2004.
[36] G. Shani, D. Heckerman, and R. I. Brafman, "An MDP-based recommender system," *J. Mach. Learn. Res.*, vol. 6, pp. 1265–1295, Sep. 2005.
[37] D. Bokde, S. Girase, and D. Mukhopadhyay, "Matrix factorization model in collaborative filtering algorithms: A survey," *Procedia Comput. Sci.*, vol. 49, pp. 136–146, 2015.
[38] F. Yao-Ning, G. Yun-Fei, and D. Xue-Tao, "An improved regularized singular value decomposition recommender algorithm based on tag transfer learning," *J. Electron. Inf. Technol.*, vol. 35, no. 12, pp. 3046–3050, 2013.
[39] H. Ma, H. Yang, M. R. Lyu, and I. King, "SoRec: Social recommendation using probabilistic matrix factorization," in *Proc. 17th ACM Conf. Inf. Knowl. Manage.*, 2008, pp. 931–940.
[40] H. Zhang, "A solution of multidisciplinary collaborative simulation for complex engineering systems in a distributed heterogeneous environment," *Sci. China F, Inf. Sci.*, vol. 52, no. 10, pp. 1848–1862, 2009.
[41] D. Kavetski, P. Binning, and S. W. Sloan, "Adaptive backward Euler time stepping with truncation error control for numerical modelling of unsaturated fluid flow," *Int. J. Numer. Methods Eng.*, vol. 53, no. 6, pp. 1301–1322, 2002.
[42] S. Eckert, H. Baaser, D. Gross, and O. Scherf, "A BDF2 integration method with step size control for elasto-plasticity," *Comput. Mech.*, vol. 34, no. 5, pp. 377–386, 2004.
[43] J. R. Dormand and P. J. Prince, "A family of embedded Runge-Kutta formulae," *J. Comput. Appl. Math.*, vol. 6, no. 1, pp. 19–26, 1980.
[44] S. P. Norsett, "An A-stable modification of the Adams-Bashforth methods," in *Proc. Conf. Numer. Solution Differ. Equ.*, 1969, pp. 214–219.
[45] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems Second Revised Edition With 137 Figures* (Springer Series in Computational Mathematics), vol. 14. 1996.
[46] G. Söderlind and L. Wang, "Adaptive time-stepping and computational stability," *J. Comput. Appl. Math.*, vol. 185, no. 2, pp. 225–243, 2006.
[47] G. Hall, W. H. Enright, T. E. Hull, and A. E. Sedgwick, "Detest: A program for comparing numerical methods for ordinary differential equations," Dept. Comput. Sci. Technol., Univ. Toronto, Toronto, ON, Canada, 1973.

JIAXIN ZHAO received the B.S. degree in automation from the Beijing Institute of Technology, China, in 2013. She is currently pursuing the Ph.D. degree in control science and engineering with Tsinghua University, China.

Her research interests include multidisciplinary modeling, numerical analysis, and data mining for complex simulation processes.

HONGWEI WANG received the B.S. degree in information technology and instrumentation from Zhejiang University, China, in 2004, the M.S. degree in control science and engineering from Tsinghua University, China, in 2007, and the Ph.D. degree in design knowledge retrieval from the University of Cambridge.

From 2011 to 2018, he was a Lecturer and, then, a Senior Lecturer in engineering design with the University of Portsmouth. He is currently a Tenured Associate Professor with Zhejiang University and the University of Illinois Urbana–Champaign Institute. His research work, in these areas, has led to the publication of one monograph and over 90 peer-reviewed papers in well-established journals and at conferences. His research interests include the application of intelligent and computing technologies to address specific design issues, for example, knowledge and information management, collaborative product development, collaborative modeling, and simulation and sustainable design.

HEMING ZHANG received the Ph.D. degree in mechanical engineering from Zhejiang University, China, in 1995.

From 1996 to 1997, he was a Postdoctoral Researcher with the National CIMS Engineering Research Center (CIMS-ERC), Tsinghua University, where he has been a Faculty Member, since 1998. He is currently a Full Professor and the Director of the Collaborative Design and Simulation Laboratory. His research interests include modeling and simulation of complex systems, intelligent manufacturing, and industrial big data technologies. His research work has led to the publication of over 100 papers on the fundamental theory, technology development and engineering applications of multi-scale modeling, simulation-based reasoning techniques for complex systems, and multidisciplinary collaborative simulation based on heterogeneous CAE systems.