3.3 A Matlab-Based Dynamic TEM Simulator

Arturo Tejada, Arjan J. den Dekker and Paul MJ van den Hof

Delft Center for Systems and Control, Delft University of Technology, The Netherlands

The Need for a Dynamical Simulator

Next generation transmission electron microscopes (TEMs) will become automated measurement tools rather than image generating devices. They will be specifically designed to extract information from specimens, like particle size distribution, chemical composition and structural information. Therefore, future electron microscope designs should take into account electro-mechanical requirements and particularly the sensing and actuation requirements needed to implement the feedback or feed forward control loops necessary for automation. They must also consider the coordination between the constituent parts, so that automated procedures may be executed.

A software-based simulator would be a helpful and cost-effective tool to study different design alternatives, providing insight into what to design and how to design it. It needs to take into account the dynamical properties of the components of interest and allow easy testing of different closed-loop control ideas. Such a simulator should be capable of recording the temporal responses of components. Also, in the ideal case, it will simulate the video stream that the microscope would generate under the observed component responses. Eventually, the simulator should provide insight into the way current components and processes can be re-utilised or reconfigured to aid in the automation process.

Implementing such a simulator requires an understanding of how the different internal components interact when setting particular variables of interest. For instance, the amount of defocus is set by the current applied to the objective lens, perturbed by the position of the specimen in space which, in turn, is determined by the specimen holder and measured from images [1, 2]. In addition, it requires an understanding of the components' temporal (i.e. dynamic) properties, e.g. their reaction speed, and knowledge of their temporal sequencing, i.e. the order/timing of component operations.

Implementing such a simulator for a TEM was started using Matlab's Simulink tools. Its architecture and capabilities are summarised next.

Simulator Architecture and Capabilities

The first step towards developing the simulator was to provide a model of the functionalities of current TEM components and their relations from the point of view of a control engineer. Figure 3.12 shows an example of such a model.

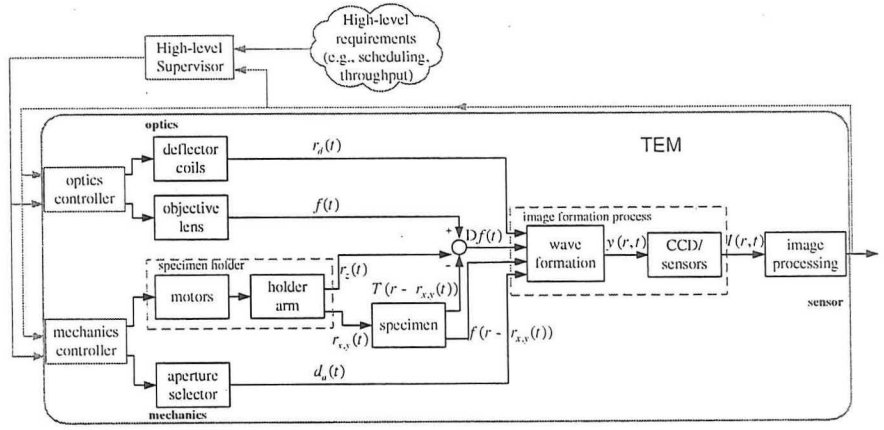

Fig. 3.12 Functionalities of current TEM components from a control engineering perspective

The boxes indicate components or processes whose dynamical models must be identified from first physical principles or through experiments [1]. For some of these boxes the models are known, or have been identified (see Sects. 7.2 and 7.3).

Note that as long as these models are not yet available, for simulation purposes and testing the unknown models can be replaced by a sensible default behaviour, e.g. a constant gain.

Additionally Fig. 3.12 shows a two-tier control structure. The lower tier contains an array of local controllers in charge of regulating individual components or processes. The higher tier contains a high-level supervisor in charge of temporal and sequential coordination of the different components. It is also responsible for trade-offs between accuracy and performance, etc. The main difference between the supervisor and the local controllers is that the former generally displays finite dynamics. That is, in general it cannot be described by a differential or a difference equation. Instead, its dynamics are described by an automaton (see [6] and the next subsection). Finally, note that Fig. 3.12 also shows processes that are not easily described by either differential equations or automatons. These include, for instance, the image formation process, which is a statistical, time-varying counting process [3], and the image processing algorithms needed to estimate parameters of interest from images.

The second step towards the simulator development is to perform a timing analysis. Note from the above discussion that some components operate continuously over time (e.g. the objective lens). Others operate only when they are triggered by an event (e.g. the image processing algorithm is executed only when an image is available). Operation of the first type of components is simulated by time-discretizing the differential equations that model their behaviour, using a small time step related to the component's dynamics, relevant from a system's perspective [4]. On the other hand, operations of event driven components are generated by the high-level supervisor whose associated automaton in turn triggers

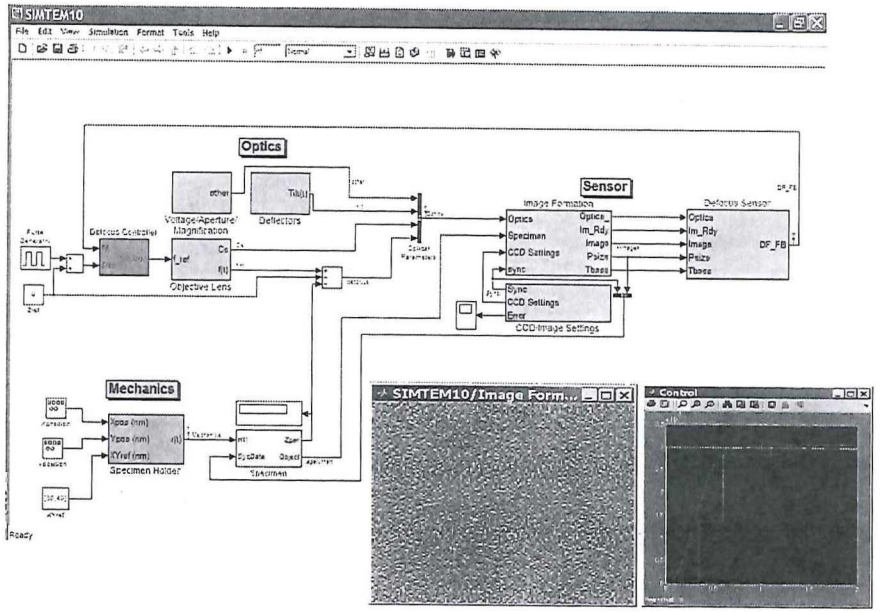

Fig. 3.13 Simulink implementation of the TEM dynamic simulator

the execution of function blocks. The function blocks themselves are used to execute complex Matlab code, such as image analysis, without affecting the simulation time (see [5] for details).

Once the dynamical models are discretised and the structure of the supervisor's automaton specified, the simulator can be directly implemented in Matlab's Simulink, as shown in Fig. 3.13. The simulated components have been interconnected following the architecture given in Fig. 3.12. This illustrates the simulation of current TEMs, allowing validation of the components and the current system behaviour. Adding the supervisor and advanced control algorithms (see Sects. 6.2, 6.3 and 7.3) allows the use of the simulator in forward engineering, to provide early validation of new concepts and implementations. The functionalities currently available in this simulator include [7]: live display of TEM internal signals (e.g. defocus, specimen position, etc.); 'live' image stream; online adjustment of magnification, defocus, beam tilt, specimen position, camera integration time, image rate, gun voltage, and camera pixel size; offline adjustment of image size; online defocus measurements; active coupling of the specimen's topography with the defocus level; and online defocus control.

Note that the temporal coordination between the continuously operating components and the event-driven component requires special attention. Although this coordination can be attained using ad-hoc methods and good engineering intuition, it is better to use systematic methods such as those in [6] to avoid developing a difficult-to-troubleshoot simulation. The next subsection provides some insight on these issues.

On the Coordination of Continuous and Event-Driven Dynamics

The events that trigger the response of event-driven components are issued by the high-level supervisor, which is in turn modelled by an automaton. The simplest automaton model is a finite state machine (FSM). Note that there is no timing information associated with an FSM. In practice, however, the input values do appear at particular times. If the time step used to time-discretise the differential equations is small enough (see previous subsection), it can be assumed with small approximation error that the input values appear at integer multiples of the time step. This allows one to treat an FSM as a very specific type of discrete-time system whose state changes every 'tick' of a clock (with a period equal to the time step) and to implement it in Simulink.[2]

Note that FSMs can also be used to aid in the temporal coordination of components, since they can divide the clock rate or provide counting mechanisms to trigger certain actions only after a certain number of clock periods (see e.g., [9]).

Finally, as stated in the previous subsection, some processes in a TEM cannot be easily modelled via differential equations, automata or FSMs. For instance, the image formation process involves counting electrons, each of which has an associated equation of motion that depends continuously on the microscope's time-varying optical parameters. The flight time of each electron is in the nanosecond order, which is about 1 million times shorter than the time scale of the main TEM components. Furthermore a large number of electrons is needed to form an image.

It is obvious that concurrently simulating the dynamics of the image formation process and those of the main TEM components is not a sensible way to go. The solution to this problem is to simulate these processes offline. For instance, to simulate the image formation processes the values of the optical parameters during a period of interest are provided to the off-line simulation, including the period of interest (e.g. 10 time steps). This data is then used by the function block to simulate and aggregate different image slices (one per time step), by approximating the optical parameters' variation with piecewise constant functions and using statistical image models (see [1, 3]). Note that, from the point of view of the simulator, the image slices are created offline in zero time (the Matlab function block consumes zero simulation time). Thus, the total image simulation time is equal to the period of interest (10 time steps in this example). In case additional delays are required (e.g. to simulate the CCD camera read-out time), they can be easily added to the simulator.

[2] Details on how to transform a direct graph into a discrete-time LTI system in Simulink can be found in [8].

Conclusions

Although the availability of a TEM simulator is very helpful, its realisation and validation turned out to be much more complex and required more effort than expected. Many of the necessary models, of both the current components as well as the image formation processes, were not readily available, or only available as theoretical models that were not tuned to the specific electron microscope involved. However, the overall modelling, to the level that implementation of the simulator became possible, created helpful insights that would have been difficult to obtain otherwise.

References

1. A. Tejada, A.J. Den Dekker, W. Van Den Broek, Introducing measure-by-wire, the systematic use of systems and control theory in transmission electron microscopy. Ultramicroscopy **111**, 1581–1591 (2011)
2. A. Tejada, W. Van Den Broek, S. Van Der Hoeven, A.J. Den Dekker, Towards STEM control: modelling framework and development of a sensor for defocus control, in *Proceedings of 48th IEEE Conference on Decision and Control*, (2009) pp. 8310–8315
3. A. Tejada, A.J. Den Dekker, The role of Poisson's binomial distribution in the analysis of TEM images. Ultramicroscopy **111**, 1553–1556 (2011)
4. G.F. Franklin, D.J. Powell, M.L. Workman, *Digital Control of Dynamical Systems*, 3rd edn. (Addisson Wesley, Menlo Park, 1998)
5. Mathworks, Using the matlab function block (2011), http://www.mathworks.nl/help/toolbox/simulink/ug/f6-6010.html
6. C.G. Cassandras, S. Lafortune, *Introduction to Discrete Event Systems*, 2nd edn. (Springer, New York, 2011)
7. A. Tejada, Measure by wire (2011), http://www.tejadaruiz.net/MBW/html
8. A. Tejada, Stability analysis of Markov jump linear systems with Markov inputs, Dissertation, Old Dominion University, Virginia, (2006)
9. R.J. Tocci, N. Widmer, G. Moss, *Digital Systems: Principles and Applications*, 11th edn. (Prentice Hall, NJ, 2011)
10. A. Tejada, J.R. Chávez-Fuentes, P. Vos, Stability and performance analysis of dual-rate systems with random output rate via markov jump linear system theory, in *Proceedings of the 50th IEEE Conference on Decision and Control*, Orlando, Florida, pp. 2895–2900

Chapter 4
Software Architecture

Abstract Specific models of the software architecture are required to predict the behaviour of a system. Unfortunately, only a limited number of accurate models are available, and no generally accepted modelling paradigms exist for many of the relevant aspects (especially for run-time execution and threading). One of the aspects considered problematic in the electron microscope system investigated was the large amount of threads created at run-time. The system was reverse engineered to model the thread usage, with the goal to reduce this amount significantly. This chapter describes the development of a modelling approach for this purpose, starting from the well-known viewpoint approach. Furthermore, it describes the application of this method to the electron microscope and a software system for validating the suitability of the approach.

Keywords Software architecture · Modelling · Concurrency · Parallelism · Architectural viewpoint · Thread pool · Performance bottleneck

4.1 Thread Performance Modelling

Naeem Muhammad and Yolande Berbers

DistriNet, University of Leuven, Belgium

Introduction

Developing high quality software is hard. Software Quality Attributes have been introduced in the field of software engineering to enable measuring the fitness and