REAL-TIME FURNACE TEMPERATURE EMULATION: A MATLAB-SIMULINK/RTW/xPC-embedded project

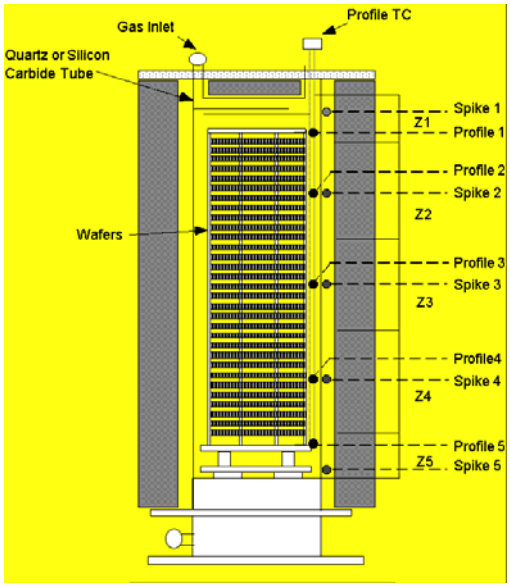
Kostas S. Tsakalis
July 17, 2003

Abstract

The real-time workshop (RTW) of SIMULINK and the xPC Toolbox offer exciting capabilities for the design and construction of embedded projects. These can be used for R&D, rapid prototyping, as well as education purposes. In this note, we discuss the development of an educational project in control systems.

Introduction

This project is motivated by the temperature control problem of diffusion/CVD furnaces, used in semiconductor manufacturing. In a quick description, these furnaces are long (1-2m) insulated tubes heated by electric heating elements. There is a great variety of furnaces in terms of construction, dimensions and physical properties: they are made of quartz or silicon carbide; their orientation can be vertical or horizontal; they can process 50-300 wafers at a time; the wafer diameters can range between 3in and 12in in diameter; and they can contain 3-5 heating zones. Temperature information is typically provided by two sets of thermocouples, one outside the electric heating tube next to the heating element (spike) and the other inside the tube (profile). Each set of thermocouples measures the temperature at different points along the tube, roughly corresponding to each heating zone.



The control objective is to maintain a uniform temperature distribution at all times inside the furnace (zone matching) according to a set-point command that is specified by the processing recipe. The set-point undergoes ramp-up/down changes that must be followed with minimal overshoot. Disturbances that affect the temperature across the furnace may be due to heat losses, gas flow, and loading/unloading operations. Furthermore, although the temperature uniformity requirements are most stringent during processing, a uniform temperature distribution is desirable at all times so that all wafers undergo identical operations. Finally, due to throughput maximization requirements, it is desired that the transient response converges as quickly as possible during set-point changes.

To solve this problem an inner-outer loop hierarchical controller structure is adopted, conforming with the two intended modes of operation, namely spike control and profile control. In this structure the controller is formed as a cascade of two subsystems where the inner one (spike controller) performs the spike temperature control and, depending on the mode of operation, receives commands from the processing recipe or from the outer-loop controller. The outer subsystem (profile controller) performs the profile temperature control, by receiving set-point commands from the processing recipe and supplying spike temperature commands to the spike controller. Depending on the furnace, there may be various degrees of coupling between input powers and temperature outputs across the zones. For this reason, a multivariable controller design can offer significant performance improvements over the more traditional decentralized approach.

On the other hand, embedded controllers using PC-104 (or similar) boards offer advantages in standardization of the computing platform and great flexibility in the design of the control system. One issue arising in the implementation of such a controller is a series of quality-control tests of the software and hardware must be performed to ensure that the controller functionality and communication software operate according to specifications. While many of these tests can be performed in a simulated environment, an implementation on an actual (and expensive) furnace is, of course, indispensable.

Motivated by this problem, we decided to develop a real-time emulator of a furnace exhibiting most of the characteristics of real furnaces as seen by an embedded controller. In particular, the furnace emulator must operate in real-time, provide analog and digital outputs, receive analog and digital inputs and exhibit largely similar behavior to a production furnace on a global scale. To achieve these objectives we chose to work with a PC-104 embedded controller board as our hardware platform, running SIMULINK's xPC-embedded Toolbox. The real-time capabilities of the xPC Toolbox are well-suited to our requirements, while SIMULINK's environment allows the implementation of fairly sophisticated models. Furthermore, the PC-104 platform is very compact and any lessons learned are directly transferable to embedded system implementations. On the other hand, for the simulator we chose to develop a simplified first-principles model that describes the heat transfer during typical operations as well as during boat-push. Finally, for the communication between the furnace emulator and the controller, (another PC-104 embedded controller, or -in our case- a notebook PC running MATLAB/SIMULINK) we used the RS-232 serial port. While in industrial systems the controller is typically sampling the furnace thermocouples directly, there are cases of furnaces using a separate data acquisition board, communicating with the controller via a serial port. Nevertheless, with the addition of an A/D-D/A board our PC-104 solution is able to produce real-time analog signals that emulate the behavior of thermocouples, except their notorious loading effects.

It should be emphasized at this point that the final result of this project can be viewed as a "virtual furnace" that, in addition to its usefulness in the original furnace control problem, it can serve as a flexible educational tool in the design and implementation of embedded control systems. The implementation details discussed subsequently should also provide an indication of the development time and the model-size capabilities of the platform. It should be kept in mind that one of our objectives during this work was to test the "rapid prototyping" concept with SIMULINK, keeping the development of custom code to a minimum. In the same vein, it is important to comment on the expertise requirements from the user. For this purpose, the author represents a fairly good "Guinea-pig," having reasonable expertise in basic MATLAB v.4 coding, but mostly "encyclopedic" knowledge of v.6 (R13) details. The main resource used was MATLAB's on-line help. It leaves much to be desired, it contains mostly trivial examples, but proved to be adequate nonetheless.

The rest of this note contains the development, assumptions and simplifications of the first-principles model and an illustrative sample run. Note that details on hardware/software information, creation of the standalone embedded code, and communication issues are now supplied in the companion document "NOTES ON BUILDING REAL-TIME APPLICATIONS WITH MATLAB" in the file ExperimentNotes.doc.

Development of the first-principles model.

The heat-transfer model for the furnace was developed in an earlier project to provide a global simulator for the testing of design procedures of embedded control systems. The model is simplified to run in real-time but contains sufficient detail for the results to emulate the behavior of actual furnaces in their entire range of operation. One of the key simplifications is the assumption of one-dimensional flow and heat transfer and lumped parameters across thin slices (disks). Such a model lacks the ability to describe temperature variations across a wafer but exhibits fairly reasonable similarity to industrial furnaces, in terms of real-time measurements (spike-profile temperatures). Another feature of this model is the ability to describe the transient behavior during loading/unloading operations (boat-push).

The model development begins with the heat balance equations including radiation and convection terms. In our first simplifying assumption, we consider that the heat transfer occurs between thin slices of the furnace where temperature is described by a few lumped parameters, one for each component of the furnace. That is, there are four components in a furnace: the heating element, the quartz tube, the gas flow and the wafer boat. In this, the heat losses to the ambient through the tube insulation are neglected and the convection between element and tube is treated as quasi-steady state due to its low time constant. On the wafer side, the most limiting assumption is that of uniform (plug) flow. This allows the problem to be treated as an one-dimensional heat exchange, but it is not a very realistic assumption (at least on the outset). Here, the fact that the primary interest is on the measurements as seen by the controller provides an excuse to ignore these effects. (Execution speed is also a concern to further justify this simplification.) In addition to that, for the radiation heat transfer only, the wafer boat is treated simply as a cylinder, avoiding the computation of fairly involved expressions.

Finally, the furnace model accounts for a showerhead and a pedestal heat masses. In both of these, in addition to radiation heat exchange with the quartz tube and the wafer boat, there is convection/radiation heat losses to the ambient. To facilitate vectorization of the variables, the temperatures of these two elements are grouped with the wafer boat temperature (but treated separately). Thus, the model contains 4n state variables, where n is the number of slices used for spatial discretization. Ideally, n should be large enough so that each slice contains no more than one wafer. But MATLAB and the compiler break down for n = 76, corresponding to two wafers per slice (see subsequent discussion). Nevertheless, with the more coarse discretizations the end result looks reasonable and adequate for our purposes.

Radiation provides the more significant contribution [ref] but convection is not always negligible especially during transients. The convection term is fairly standard (hA[T1-T2]) where an empirical nonlinear expression is used for the heat transfer coefficient h [Perry]. Around the wafers, the convection term is computed with different coefficients for the top and bottom surfaces, depending on whether the wafer is heated or cooled. It is probably a minor detail (at least by looking at the simulation results) but its Matlab coding is fairly straightforward and easily vectorized. The model for the radiation term is a lot more complicated and interesting. The usual expression (FAs[T41-T42]) requires the computation of the view-factors F between two surfaces. The integral expressions for cylinder and disk surfaces are easily computable off-line and stored as part of the problem data. The radiation term for the element-tube heat exchange can be computed in a vectorized form by a matrix multiplication. The main issue here arises from our desire to model the boat-push process. During boat push the view factors between the element/tube slices and the wafers are changing and the view factor matrix needs to be re-computed each time instant. There are two possibilities for this: The more elegant one involves the construction of an s-function, but requires some expertise on C-coding and the use of pointers, which is how MATLAB/SIMULINK communicates with s-functions. The other is a “brute-force” interpolation or switching among several matrices, one for each location of the boat. The latter method increases the problem data considerably (each matrix has dimensions 2n x 2n) but can be easily implemented with standard Simulink blocks.

Beyond this core, the model contains three grouped subsystems. One performs the various input computations, unit conversions and translation of the boat location indicator (0-1 in “human-friendly” dimensionless units) to a discrete absolute location. Another subsystem performs the computation of the outputs, i.e., thermocouple temperatures at five locations. This computation involves a weighted sum of the entire state vector with weights computed based on (simplified) view factors. The modeling of fast thermocouple dynamics was avoided; instead, a linear first-order low-pass filter was included after the thermocouple reading computation. The final subsystem introduces simulated noise, quantization effects, and TC nonlinearity.

The top layers of the Simulink code are shown in Fig. 1-3.

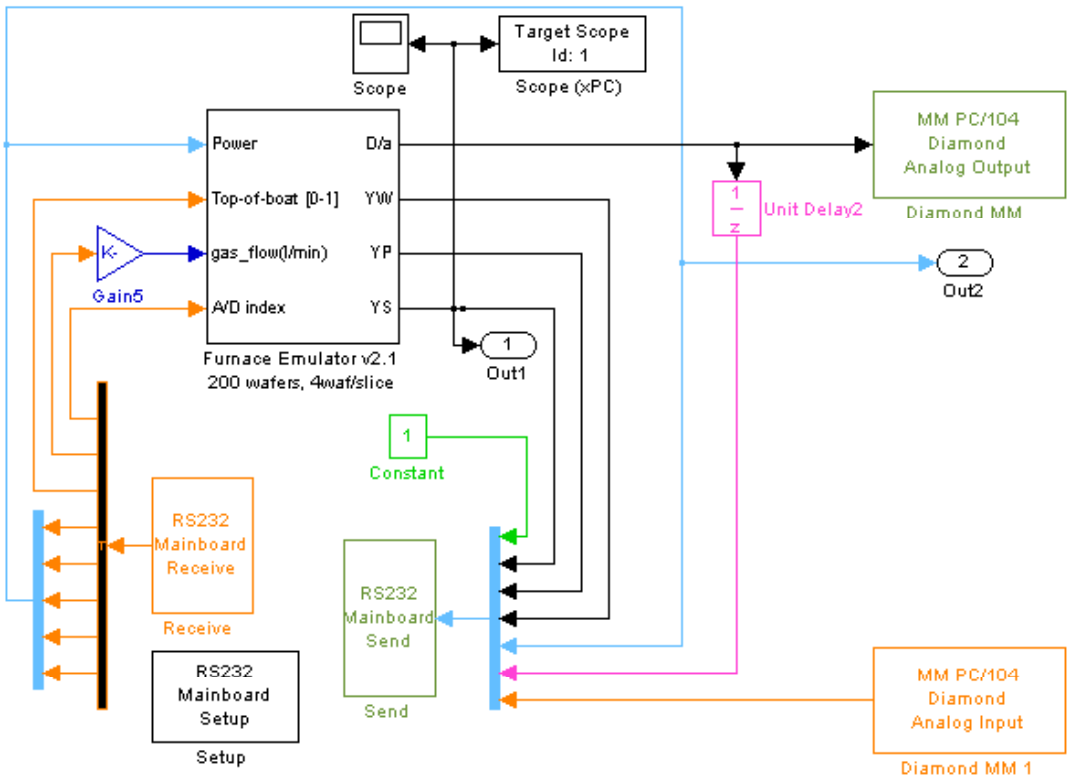


Figure 1: Block diagram of the Simulator top layer

K. Tsakalis, 11/02
kst@asu.edu

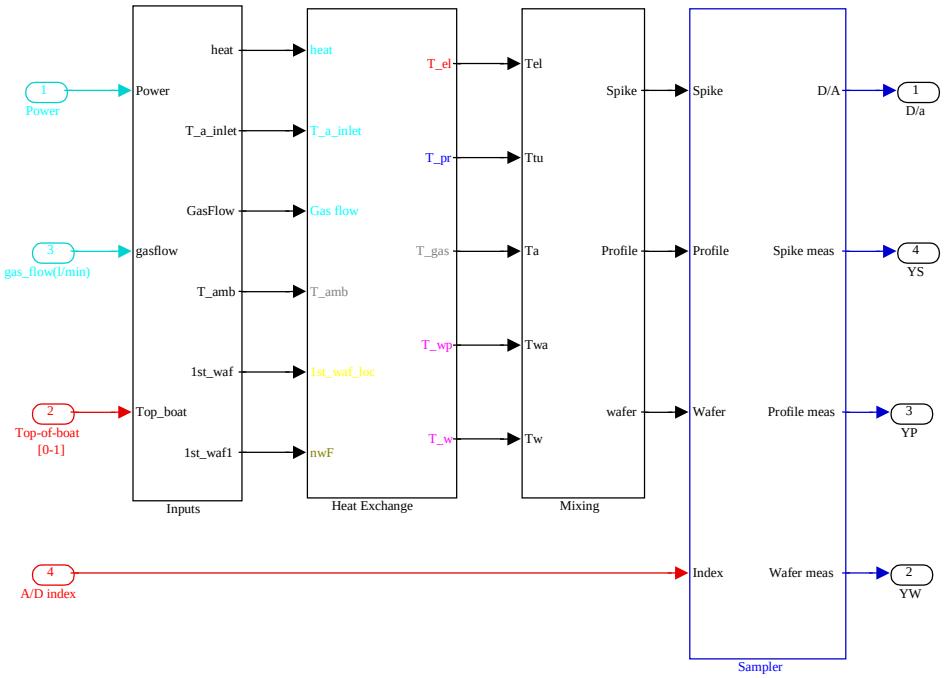


Figure 2: Block diagram of the Simulator second layer

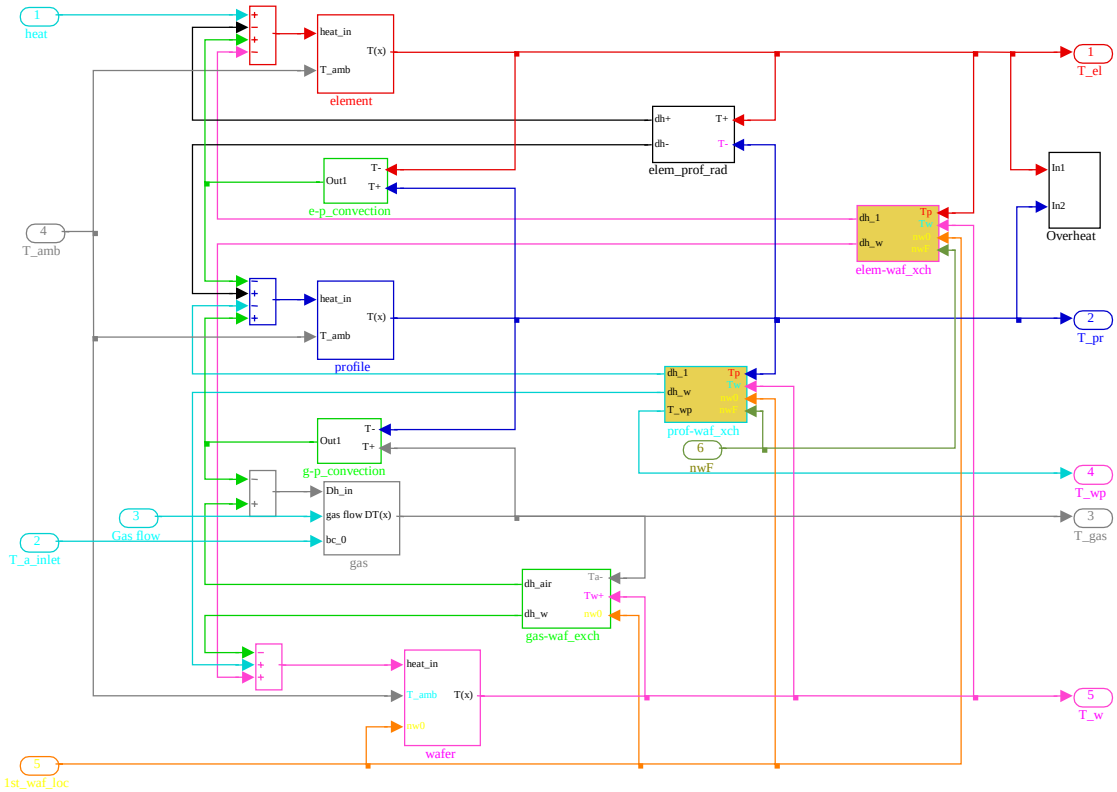


Figure 3: Block diagram of the heat exchange subsystem

Sample Run Results

After the successful completion of the build process, the simulator is downloaded and executed on the target (PC-104). Before the communication with the host PC is initiated, the RS-232 serial inputs are initialized to 0. Then, the host controller is initialized and the simulation is started (the serial port should be initialized only once per session). Since the host controller waits for a message to be received before computing and transmitting the next input, the execution is real-time, controlled by the execution of the simulator on the target PC-104. In this sequence, it is necessary that all controller computations are completed before the new message is received. While this does not present a problem in terms of net computational speed, it may not be the case if a new Windows process is started. In such an event, a sample may be missed or received out-of-sync but this will be handled by the given function and the communication will be re-initialized to receive the current sample at the next sampling instant. Using a simple mechanism to detect such an event, the effect on closed-loop performance is negligible, as long as samples are missed only occasionally. More elaborate mechanisms can preserve performance even with longer intervals of missing data (e.g., using observers) but this does not seem to be of concern in this application.

The serial communication with the host presented many problems, likely to be caused by the Windows operating system and/or the MATLAB access of the serial port. In general, most of these problems occur when writing to the port and the simulation becomes very sensitive to background processes and the state of the operating system. (They should disappear, or at least be minimized, when the controller is also implemented in a dedicated PC-104 platform and the laptop is delegated the responsibility of only collecting and displaying data.)

After the end of the run, the controller should be stopped first otherwise the communication function spends most of its time looking for transmitted data with a 5sec timeout, and stopping the simulation could be difficult. Then the target should be switched off or the laptop should be shut down. If MATLAB is closed before the target is shut down, Windows encounters a fatal error.

Notice that, if the serial communication fails during the run, each variable maintains its last value. This is a minor inconsistency with an actual furnace where failure to communicate with the controller would return all variables to the 0 default values, to avoid overheating. It is possible to adjust the behavior of the furnace simulator during communication failures but this would require extensive coding.

The results (applied power, temperatures) of the first few minutes of the first sample run are shown in Fig. 7. In this run, the furnace simulator execution was controlled by MATLAB with the xpcrctool function. The continuous communication between the host and the target using the ethernet connection does not appear to cause any major interference with the timing of the serial read/write operations. At approximately 1.5 min after the start of the control, there were several Windows operations performed on the laptop intentionally to cause the loss of communications timing. This event was handled without problems. The model had 4x59 states for the heat transfer block and used an 10 discretization points in the view-factor computations, depending on the location of the boat. It executed at an average execution time of 0.07 sec, never exceeding 0.1 sec.

In terms of the physical process, the simulation shows the behavior of the furnace starting from an idling condition with spike temperatures at 500 deg. The controlled variable is the spike temperature with a set-point of 600 deg. The boat of wafers is loaded six minutes after the start of the simulation. At that point the profile temperatures drop due to the large cold mass entering the furnace and the wafer temperatures sart to rise as the boat moves inside the furnace. The loading operation is completed in 2 min. During that period the spike temperature fluctuation is less than 5 deg. This is not surprising since the applied power has a much larger effect on the element temperature than the conditions inside the tube. On the other hand, the modern industrial practice of controlling the profile temperatures is a considerably more difficult task.

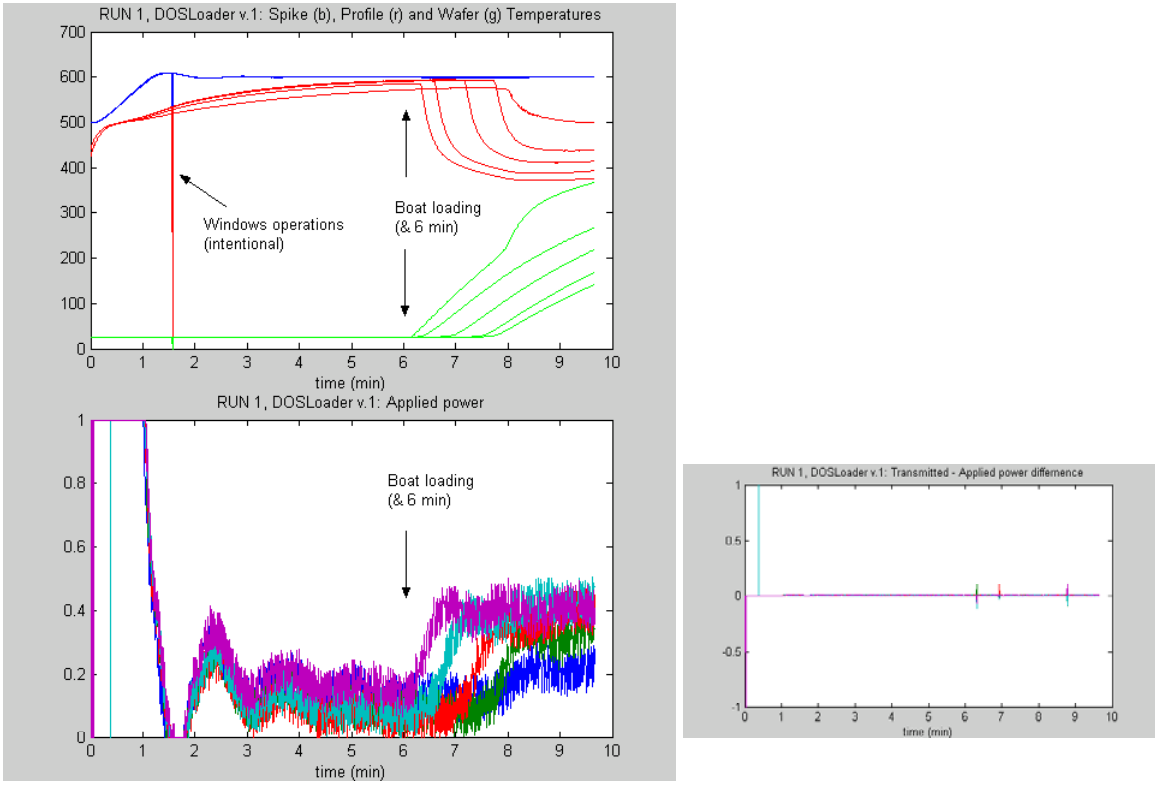


Figure 4: Temperature and power data from the first sample run.

The results (applied power, temperatures) of the second sample run are shown in Fig. 8. Again, the furnace simulator execution was controlled by MATLAB with the xpcrctool function. In this case the code was executed for about one hour and there are several instances of missed or mis-timed data. All except the first were unintentional and it is not known if they are caused by Windows operations or by the use of the xpcrctool function. As seen by the temperature and power plots, mis-communication instances occur at both the transmission and reception ends. All events were handled without major problems. While mis-communication of the temperature data in the controller has a virtually negligible effect, this is not the case for mis-communication of power data to the furnace. An one-sample application of full power at steady-state causes a temperature ripple of about 0.5 deg in the spike temperatures. Of course, this would be unacceptable for a real furnace and a more elaborate error handling procedure at both ends of the communication process would be required for industrial applications. As in the first sample run, the model had 4x59 states for the heat transfer block but used the custom s-function block to perform the view-factor computations, depending on the location of the boat. It executed at an average execution time under 0.06 sec, never exceeding 0.1 sec.

In terms of the physical process, the simulation shows the behavior of the furnace starting from an idling condition with spike temperatures at 500 deg. The controlled variable is the spike temperature with a set-point of 600 deg. The boat of wafers is loaded six minutes after the start of the simulation. At that point the profile temperatures drop due to the large cold mass entering the furnace and the wafer temperatures sart to rise as the boat moves inside the furnace. The loading operation is completed in 2 min. At 13 minutes the set point is ramped-up to 900 deg, at 250 deg/min (this ramp rate is not achievable by the furnace, hence the considerable separation of the spike temperatures). At 35 min, the set point is switched back to 600 deg.

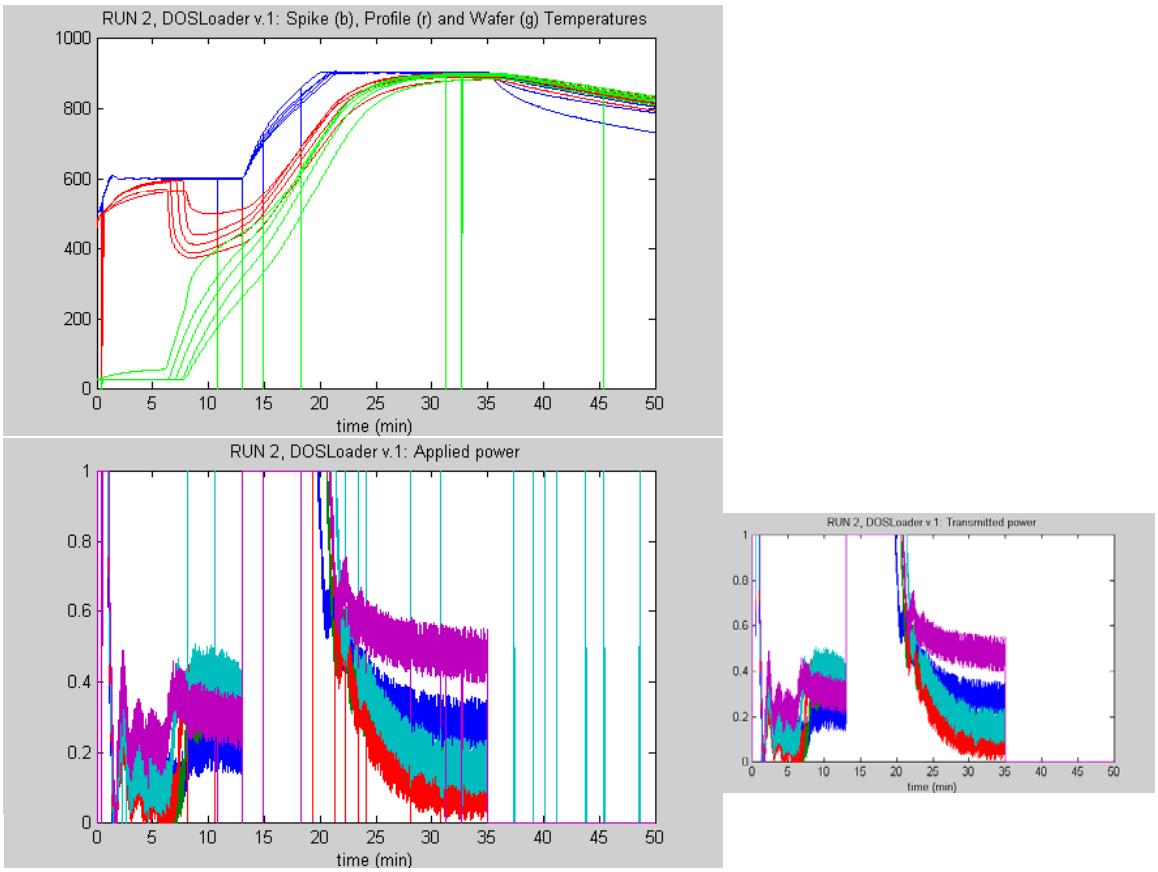


Figure 5: Temperature and power data from the second sample run.