```
process AIRPLANE
     call TOWER giving GATE yielding RUNWAY
     work TAXI.TIME (GATE, RUNWAY) minutes
     request 1 RUNWAY
     work TAKEOFF.TIME (AIRPLANE) minutes
     relinquish 1 RUNWAY
end " process AIRPLANE
```


```
process AIRPLANE
     call TOWER giving GATE yielding RUNWAY
     work TAXI.TIME (GATE, RUNWAY) minutes
     request 1 RUNWAY
     work TAKEOFF.TI
     relinquish 1
end " process AI
```


Simscript II.5

Since 1962

Introduction to Combined Discrete-Contnuous Simulation Using SIMSCRIPT II.5

Abdel-Moaty M Fayek

Computer Science Department
California State University, Chico

CACI

If there are questions regarding the use or availability of this product, please contact CACI at any of the following addresses:

For product Information contact:

CACI Products Company
1011 Camino Del Rio South, suite 230
San Diego, California 92108
Telephone: (619) 542-5224
www.cacias1.com

CACI Worldwide Headquarters
1100 North Glebe Road
Arlington, Virginia 22201
Telephone: (703) 841-7800
www.caci.com

For technical support contact:

Manager of Technical Support
CACI Products Company
1011 Camino Del Rio South #230
San Diego, CA 92108
Telephone: (619) 542-5224
simscript@caci.com

Table of Contents

List of Figures

PREFACE

This manual is an introduction to SIMSCRIPT II.5 continuous system simulation. The emphasis is on the combined system simulation features. The models are written in SIMSCRIPT II.5.

This is a revised version of the original manual written by Abdel-Moaty M. Fayek. The manual consists of two chapters and an appendix. The first chapter introduces the SIMSCRIPT statements, which are, used to model continuous processes or processes with combined characteristics. It also introduces the nature of continuous models as well as the differences between discrete and continuous models.

Chapter II provides a step-by-step tutorial toward building a combined discrete-continuous model. Three models are introduced. The tutorial starts with a discrete model, then continuous characteristics are gradually added to the model increasing its complexity. For each model presented, both graphical representations and SIMSCRIPT listings are provided. The complete listing of Model III is in Appendix A. Some models (such as EJECT and BOUNCE), which illustrate the usage of continuous simulation, are included in the SIMSCRIPT II.5 distribution.

Free Trial & Training

SIMSCRIPT II.5 is available exclusively from CACI Products Company. It can be sent to your organization for a free trial. We provide everything needed for a complete evaluation on your computer: software, documentation, sample models, and immediate support when you need it.

Training courses in SIMSCRIPT II.5 are scheduled on a recurring basis in the following locations:

San Diego, California
Washington, D.C.

On-site instruction is available. Contact CACI for details.

For information on free trials or training, please see:

www.cacias1.com or email simscript@caci.com

Chapter 1. Combined Discrete-Continuous Simulation with SIMSCRIPT II.5

1.1 Continuous Versus Discrete Models

Discrete simulation languages support a view of the world in which systems change their states in a discontinuous and instantaneous fashion. At some instant in time, for example, a customer arrives in a queue and at that moment the queue length is increased by one; or a server completes the process of serving a customer and changes status from busy to idle. The times at which events such as these occur are often determined by using random number generators which generate arrival times or service times from a given distribution. These models are not concerned with details of how the customer arrived in the queue or of what the server was doing during the service activity.

This concentration on the essentials of a system, and the omission of features which do not significantly affect those aspects of system behavior we are interested in, are the principles of sound modeling and effective simulation. There are, however, some situations in which it is necessary to study the behavior of part of a system continuously as time advances. There are simulation languages which concentrate on this aspect of system behavior. These are called continuous-system simulation languages, or CSSLs. CSSLs lack most of the capability of discrete languages (such as SIMSCRIPT II.5) to support discrete event modeling. In contrast, SIMSCRIPT II.5 offers continuous modeling features which allow continuous processes to be included in discrete models.

Later in this chapter we will describe these features of SIMSCRIPT II.5 in more detail. First, however, we will study the essentials of continuous simulation and the way in which the continuous and discrete parts of a model may be combined.

1.2 The Nature of Continuous Models

To understand the key difference between discrete and continuous models, we need to understand the different ways in which system states are perceived to change with advancing time.

The system clock for a purely discrete model advances from event to event, using entries in an event queue to determine the next event time. We assume that the system state is unchanged between events and changes only at event times.

The variables in a continuous model, on the other hand, are assumed to vary continuously with continuously advancing time. Continuous variables are often defined by means of differential equations. A differential equation can be thought of as an equation which defines a relationship between a continuous variable and its own rate of change. For example, a hot metal pellet at a temperature T° F in an ambient temperature of T_A° F can be assumed to cool such that T is given by:

$$dT/dt = k(T_A - T)$$

where t is time measured, say, in seconds. This equation states that the rate of cooling in °F/sec is proportional to the instantaneous difference between the pellet temperature and ambient temperature. Note that if T is greater than T_A, then dT/dt is negative (implying cooling) and if T is less than T_A, dT/dt is positive (implying heating). In SIMSCRIPT II.5, T and T_A would be defined as continuous real variables:

```
define T, TA as continuous real variables
```

and the derivative or rate of change of T would be represented by D.T, leading to the statement:

```
let D.T = K * (TA - T)
```

to represent the differential equation.

In general, continuous models are composed of a mixture of differential and algebraic equations. The algebraic equations define relationships between variables which are both continuously and instantaneously true.

For example, in an electrical circuit a voltage V may be calculated from an equation of the form:

$$V = V_{cc} - I * R$$

where V_{cc} and R are constants and I is defined by a differential equation, say:

```
D.I = V/L
```

Variables such as I (or T in the earlier example) are often called "state variables" in continuous systems terminology, and variables (such as V) defined by algebraic equations are called "auxiliary variables". To avoid confusion we will refer to "continuous state variables" and "continuous auxiliary variables" in the text.

1.2.1 Solution of Differential Equations

Differential equations are solved by a technique in which time is advanced in small steps with calculations to update the values of the continuous variables (and their rates of change) at each step. The same equations may be solved with different time steps. The approximation errors inherent in the process increase with the size of a time step, along with the speed of solution. Some methods automatically adjust their step sizes to keep the errors within acceptable tolerances. The numerical integration technique supported in SIMSCRIPT II.5 is a variable-step size fourth-order Runge-Kutta method. It is called automatically when required. The maximum and minimum step sizes used by the method can be controlled by adjusting the values of **MIN.STEP.V** and **MAX.STEP.V**. Control over the error criterion used by the method is available through **ABS.ERR.V**, and **REL.ERR.V**, which put bounds on the acceptable absolute error and the acceptable relative error, respectively.

It is possible to use an alternative integration routine by means of an assignment to the subprogram variable **INTEGRATOR.V**, as long as the rather complex interface requirements are satisfied.

The process of solving the differential equation is started by assigning initial values to the continuous state variables and the time, t, and then advancing time by the first step. Further steps follow and the process finishes when a termination criterion is satisfied. The termination criterion may take a number of forms. It may be simply a test to see if time has reached a maximum specified value (a finish time); it may require that a continuous variable exceed a set threshold value, or the current value of another continuous variable; or it may comprise a combination of these similar criteria (e.g. A and B, A or B). A continuous process is activated in SIMSCRIPT II.5 by using a statement of the form:

```
work continuously evaluating 'HEATUP' testing 'FINISH'
```

in which **HEATUP** is the name of the subprogram containing the differential equations and **FINISH** is a subprogram containing the termination criterion.

1.3 Combined Discrete-Continuous Models

In discrete simulation terms, the start-of-integration and the end-of-integration are events. This concept is the key to the interface of discrete and continuous parts of a combined model. Suppose, for example, that a conventional discrete simulation program has a simple continuous process embedded within it. This continuous process will be triggered to begin when some specific event occurs within the discrete model (e.g. **HEATER-ON** becomes **TRUE**). A continuous simulation process now starts and the continuous integration proceeds step-by-step to completion upon the satisfaction of some criterion. It is important that the step size of the integration process be controlled so as to synchronize correctly with events (i.e., so that events coincide with the end of an integration step). The variable-step integration method used in SIMSCRIPT II.5 guarantees this synchronization. The changes which occur in the discrete model at an event may modify the parameters of the continuous integration, including its termination time. SIMSCRIPT II.5 takes care of these effects and returns control to the integration process for further continuous integration until the next event in the queue is encountered. Control thus passes back and forth between the discrete and continuous parts of the model until the integration of the continuous process is completed.

Typically, in a combined continuous-discrete model of this kind, there will be several continuous processes. For example, in the case described below in which metal ingots are moved into and out of a furnace, the continuous processes representing the heating of individual ingots are transient and must be created and destroyed, as required. SIMSCRIPT II.5 provides a simple way of defining the creation and destruction of such processes. This is often difficult to achieve in conventional continuous languages.

The creation and destruction of continuous processes means that the form of the continuous system model (the set of differential and other continuous equations) may change repeatedly in a manner which does not normally occur in CSSL-based models. In particular, the number of differential equations may change. Although the integration procedure must be modified to account for these changes, it still operates much as it does in other continuous simulations. That is, at any instant in time, the entire set of currently-active differential equations is processed together, and the set of continuous state variables is treated as a continuous state vector which is updated by the integration routine. In other words, the continuous processes are not treated as separate entities to be updated one at a time. This is because the integration process, in advancing the solution of the system differential equations by a single time step from t to t + h, needs to access estimated values of other variables at intermediate times (such as t + h/2), and these references are often made across process boundaries. For example, in a system with 4 ingots, it may be necessary to repeatedly access the temperature of ingots 1, 2, and 3 in order to calculate the current rate of change of the temperature of ingot 4, and so on.

To summarize, a continuous process is described by one or more differential equations, sometimes with the addition of algebraic (or auxiliary) equations. It is also defined by a starting event and a terminating event which define the times of its creation and destruction. Multiple continuous processes are allowed, and can be created and destroyed at different times. The resulting set of differential equations is solved by a numerical integration procedure which updates the continuous state vector at each integration time step, adjusts its integration step size so as to keep errors within acceptable tolerances, and also synchronizes with discrete events including the initiation and termination of individual continuous processes.

1.4 Continuous Features of SIMSCRIPT II.5

Within SIMSCRIPT II.5, activities over time are represented using processes. A process encompasses a number of related and sequenced discrete events in simulated time. In the discrete domain all changes to variable values can occur only at these event times. A process, however, does provide a means for expressing lapses in simulated time using **`work/wait`** statements. These statements have been enhanced to incorporate the additional specification of a set of continuous differential equations and an associated logical termination condition. These differential equations must be associated with a process instance, so continuous variables exist only as attributes of a process. Continuous variables are declared as such by adding the descriptor **`continuous`** to the variable declaration:

```
processes
   every INGOT has an INGOT.TEMP
   define INGOT.TEMP as a continuous double variable
```

Since numerical precision is often a concern in numerical integration, all computation performed on continuous variables is double precision. Each continuous variable has two values associated with it: **`D.VARIABLE`**, which gives the current value of the continuous variables at the end of the current integration step, and **L.VARIABLE**, which gives the value of the continuous variable at the end of the most recent integration step. These variables are declared by the system. For example, consider the following:

```
define FURNACE.TEMP as a continuous double variable
```

The differential equations which describe the behavior of the continuous variables have the form:

```
D.VARIABLE = expression
```

In a simple example such as an oscillating mass-spring system, the model may have the following structure:

```
preamble
   .
   .
   processes include STOP.SIM, OUTPUT
   every SPRING has
      an X,
      a VEL
   define X, VEL as continuous double variables
   .
   .
end   ''preamble
main
   .
   .
   create a SPRING
   let D.vel = ft
   activate SPRING now
   .
   .
end  ' 'main
```

The integration process updates the continuous variables at the end of each integration step. Sometimes it is necessary to have access to the values of a variable at the end of the previous step as well as at the current step. This value is available through **L.VARIABLE**, and is useful in testing if a threshold was exceeded in the last step. In such cases a statement of the form:

```
if L.VARIABLE < threshold <= variable
```

can be used.

The time elapsed since the end of the preceding step is also available in the system variable **DELTATIME.V**.

In the examples in Chapter 2, a system is simulated in which ingots are loaded at different times into a furnace and heated. Models II and III in Chapter 2 treat this heating process as continuous.

A process **INGOT** defines an ingot waiting until a slot in the furnace is available. The ingot is then loaded (filed in the **FURNACE.SET**), and the heating process is initiated using a **work** statement:

```
work continuously evaluating 'HEATINGOT' testing 'HOTENOUGH'
```

HEATINGOT is a routine containing the differential equation:

```
let D.CURRENT.TEMP(INGOT) = expression
```

Note that **INGOT** is used here as a pointer variable to a specific process notice defined in **HEATINGOT**, since several process notices may exist at any given instant.

In the above **work** statement **HOTENOUGH** is a function which returns unity when the ingot has reached the desired final temperature, and zero otherwise. This function requests a test for completion of the current instance of the continuous process (an option in the **work** statement). Note that a working process can be **cancel**ed or **interrupt**ed. In this case it will have a **TIME.A** value of zero; a **resume** statement is thus equivalent to a re-**activate now**, with execution resuming at the statement following the interrupted **work**. To continue in the continuous work state, some explicit transfer of control to re-execute the **work** statement must be included in the process code.

Statistics may be gathered on the changing values of continuous variables and their derivatives using the standard SIMSCRIPT **accumulate** and **tally** statements. Any required statistical counters will be added as attributes of the process notice entity. The integration technique, however, does not directly address continuous variables by name, and it is therefore necessary to provide explicit assignments to those variables at the end of each integration step. An optional **updating** clause at the end of the **work** statement will cause a user-supplied routine to be called at the appropriate times. For example:

```
work continuously evaluating ..., testing ..., updating 'UPDATE'
```

The **UPDATE** routine should contain statements such as:

```
let VARIABLE = VARIABLE
let D.VARIABLE = D.VARIABLE
```

Although these assignments of variable values to themselves appear redundant, they do cause the compiler to generate additional statements which ensure that the data required for statistics, for traces of changing variable values, and for graphical displays (e.g., using PC SimAnimation) are gathered. The examples in Chapter 2 do not require the use of this facility, but it is used in some other example programs supplied with the system (e.g., the PILOT EJECTION MODEL, BOUNCE).

The above simple extensions to conventional SIMSCRIPT II.5 programming, along with the integration control statements described below, are all that is required to model continuous processes.

1.5 Integration Control Statements

Integration control statements perform the following operations:

- Select the integration algorithm to be used
- Set maximum and minimum step sizes for the integration
- Set absolute and relative error tolerances.

Since SIMSCRIPT II.5 has a single integration algorithm, which is used by default, it is not strictly necessary to specify the integration method explicitly. The form of the statement is:

```
let INTEGRATOR.V = 'RUNGE.KUTTA.R'
```

Other assignments can be made to **INTEGRATOR.V**, if an alternative user-supplied routine is available.

The maximum and minimum integration step sizes are set by assigning values to **MAX.STEP.V** and **MIN.STEP.V**. Default values are 0.1 and 0.01, respectively. Assigning values to ABS.ERR.V and REL.ERR.V sets absolute and relative error tolerances. Default values are 0.0001 and 0.01, respectively.

In Chapter 2 we will illustrate all the above concepts with a problem concerning a furnace, which is used to heat steel ingots. Three different versions of the program are presented. The first is purely discrete, the second and third involve the use of continuous processes.

Chapter 2. Using SIMSCRIPT II.5 for Combined Simulation—A Tutorial

The following tutorial presentation concerns a soaking pit furnace used to heat steel ingots [1, 2]. Three models will be introduced. The first model handles the entire process in terms of discrete events. In this model, the heating times of the ingots are generated randomly using a probability distribution. In the second model, the changes in ingot temperature are determined using differential equations. This model combines both discrete and continuous characteristics. Discrete events include the arrival and departure of the ingots. The changes in ingot temperature are continuously evaluated until the ingot reaches its final temperature. To simplify the modelling process, the furnace temperature is assumed to be constant. This assumption will be changed in the third model, in which a new set of differential equations is used to describe changes in both furnace and ingot temperatures.

The SIMSCRIPT programs described in this chapter are broken into subprograms, each described separately with listings and explanatory comments. A complete listing of the third program can be found in Appendix A.

2.1 Model I: Problem Statement

A steel plant has a soaking pit furnace which is being used to heat up steel ingots. The interarrival time of the ingots is exponentially distributed with a mean of 1.5 hours. If a soaking pit is available when an ingot arrives, the ingot is immediately put into the furnace. Otherwise it is put into a warming pit where it retains its initial temperature until a soaking pit is available. Figure 2-1 graphically represents the furnace.

Assuming that there are a maximum of nine soaking pits and that the ingot heat-up time is uniformly distributed in the interval from four to eight hours, simulate the heating process for 30 days (720 hours). Record both ingot waiting time and furnace utilization statistics. Schedule the arrival of the first ingot at time 0.

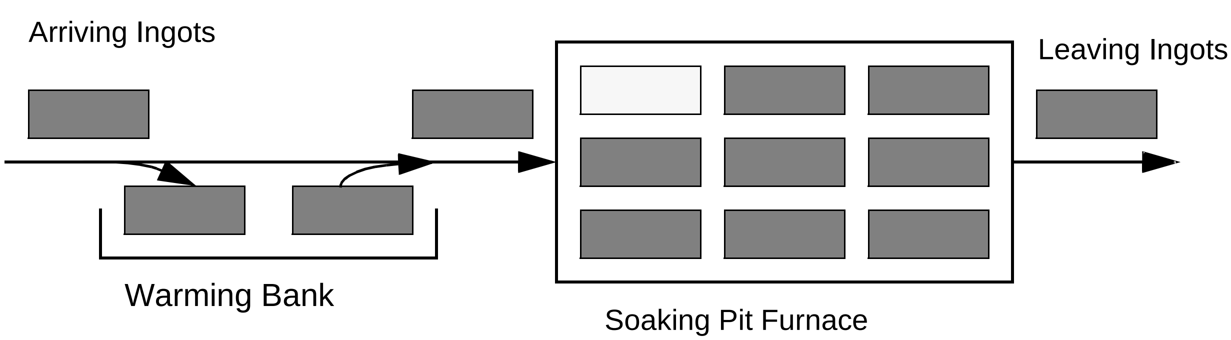

Figure 2-1. Graphical Representation of the Soaking Pit Furnace

2.2 Implementation of Model I

Model I is also the basis for Models II and III.

To construct a SIMSCRIPT model to simulate the heating process, the following are needed:

- A process to simulate the ingots
- A resource to simulate the soaking pits
- A scheduler process to schedule the arrival of the ingots
- A process which will be activated at the end of the simulation to print the final statistics and stop the simulation.

We will also use a set, **`FURNACE.SET`**, to keep track of the ingots being heated. This is not strictly needed in this model, but is included to minimize the changes needed for the following models. The block diagram in figure 2-2 illustrates the actions of the individual subprograms and their calling sequences. Use of the term sequence here does not signify a strict call-return sequence as with conventional programming languages, but refers simply to the links between a subprogram and its point of invocation.

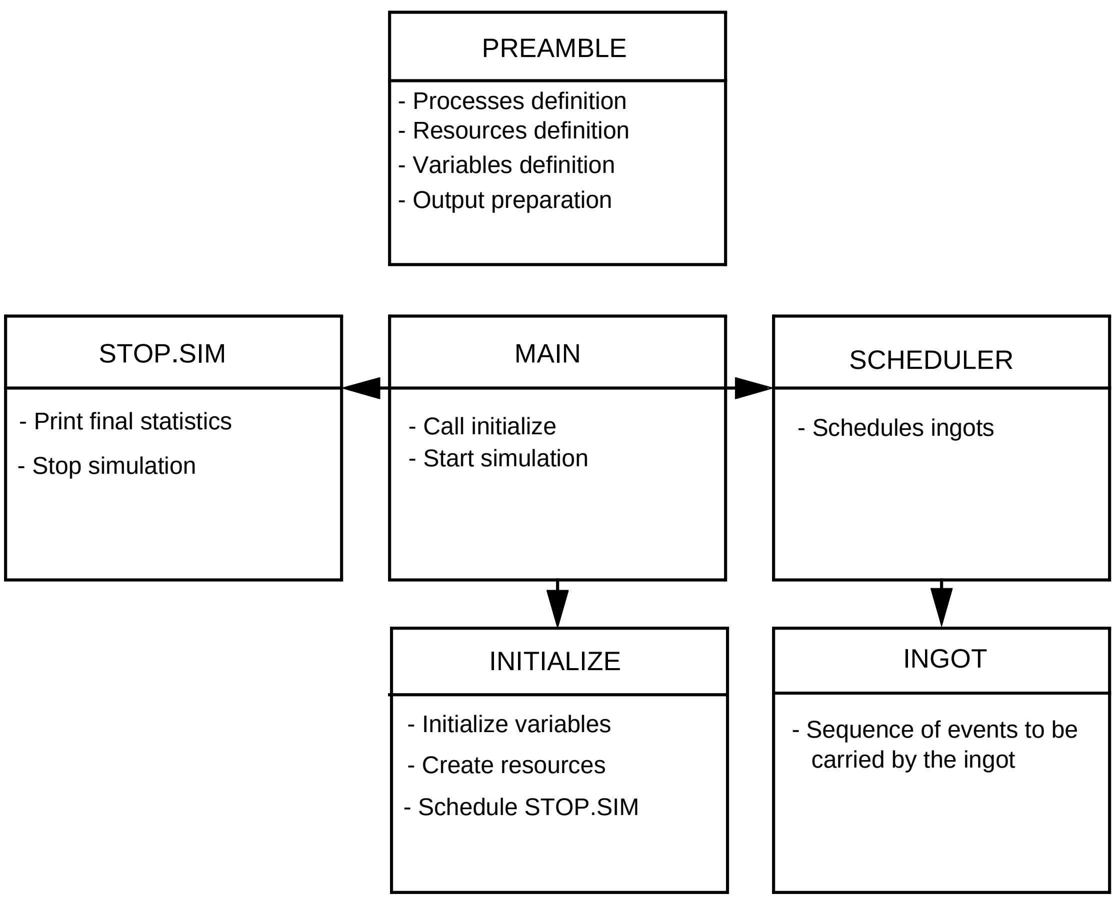

Figure 2-2. SIMSCRIPT Model I Segments

The **preamble** for this model is given in figure 2-3. It includes:

- Definition of processes,
- Global variable definition,
- Resource definition,
- Statistics performance measurement.

```
001 preamble
003   processes include STOP.SIM, SCHEDULER
004
005   every INGOT may belong to the FURNACE.SET
006   THE SYSTEM owns the FURNACE.SET
007
008   resources include PIT
009
010   define TOTAL.INGOTS as an integer variable
011   define ENDTIME as real variable
012   define HOURS to mean units ''of time
013
014   accumulate MEAN.WAIT.TIME as the mean,
015         VAR.WAIT.TIME as the variance,
016         MAX.WAIT.TIME as the maximum of WAIT.TIME
017
018   accumulate MEAN.NO.OF.INGOTS as the mean,
019         VAR.NO.OF.INGOTS as the variance,
020         MAX.NO.OF.INGOTS as the maximum,
021         MIN.NO.OF.INGOTS as the minimum of N.FURNACE.SET
022
023 end
```

Figure 2-3. Listing of the Preamble, Model I

Line 3 of the **preamble** defines the process **SCHEDULER** which schedules arrival of the ingots. It also defines the process **STOP.SIM** which is activated at the end of the simulated period to print the final statistics and signal the termination of the simulation. Lines 5 and 6 specify the fact that the ingots may belong to the **FURNACE.SET**; the **SYSTEM** is defined as the owner of the set.

The definition of the soaking pit resource pit is given on line 8. Lines 10 and 11 define the three global variables: **TOTAL.INGOTS**, which represents the number of processed ingots at any instant; **endtime**, which represents the period of the simulation; and **WAIT.TIME**, which represents the time an ingot waits to use the furnace. Line 12 defines **HOURS** as the simulation time unit. The **accumulate** statement on lines 14, 15, and 16 is used to compute ingot waiting time statistics. Another **accumulate** statement on lines 18, 19, 20, and 21 computes the furnace utilization (**N.FURNACE**. **SET** gives the number of ingots inside the furnace at any instant).

The **main** program of the model is given in figure 2-4. Line 26 calls the **initialize** routine which is described below. Line 28 calls the timing routine and begins execution of the simulation. The timing routine removes the first process notice—for the **SCHEDULER** process—from the event list. Since the activation time of process **scheduler** is now, it is scheduled at current simulation time **TIME.V**, currently equal to 0 (see line 43 of the **initialize** routine).

```
024  main
025
026     call initialize
027
028     start simulation
029
030  end
```

Figure 2-4. Listing for the Main Program, Model I

The initialize routine of the model is given in figure 2-5. It includes:

- Identification of the input file which includes the number of available soaking pits and the furnace temperature
- Initialization of the available resources
- Creation of the process **SCHEDULER**, which is used to schedule the arrival of ingots, and of the process **STOP.SIM**, which is activated at the end of the simulation period to print the final statistics and terminate the simulation.

```
031  routine INITIALIZE
032
033     open 1 for input, name is "IN.DAT"
034     use 1 for input
035
036     create every PIT(1)
037     read U.PIT(1)
038
039     read ENDTIME
040
041     activate a STOP.SIM in ENDTIME hours
042
043     activate a SCHEDULER now
044
045  end
```

Figure 2-5. Listing for the INITIALIZE Routine, Model I

Next is the process routine for process **SCHEDULER**. This routine is called each time the process notice for the **SCHEDULER** process is removed from the event list. Figure 2-6 gives the contents of this routine. This routine is called first at time zero. Upon first entry to the

WHILE loop on line 48, line 50 is executed (**activate an INGOT now**). As a result, an ingot process notice is put on the event list with an activation time equal to the current value of **TIME.V** (currently 0). The **wait** statement on line 51 places the **SCHEDULER** process notice back in the event list with activation time equal to 0 plus an interarrival time. The interarrival time, as stated on line 51, is exponentially distributed with a mean of 1.5. It is generated using random number stream 1. When the **SCHEDULER** process notice is removed from the event list for the second time, another ingot process notice is placed on the event list with another activation time.

The **SCHEDULER** process notice is again returned to the event list with activation time equal to the current value of **TIME.V** plus an interarrival time. This process will continue as long as the condition on line 48 is satisfied.

```
046   process SCHEDULER
047
048      while TIME.V lt ENDTIME
049         do
050            activate an INGOT now
051            wait EXPONENTIAL.F(1.5, 1) hours
052         loop
053
054   end
```

Figure 2-6. Listing for the Process Routine SCHEDULER, Model I

The process routine for the process **INGOT** is given in figure 2-7. This routine is called every time an ingot process notice is removed from the event list. Line 57 defines a local variable, **ARRIVETIME**, used to compute the ingot waiting time. As shown on line 59, this variable is initialized to the ingot arrival time. As the process starts, the ingot requests a soaking pit (line 61). When one is available, the ingot wait time is calculated (line 62) and the ingot is then filed in the furnace as shown on line 64. The ingot begins a heating process with a duration uniformly distributed between 4.0 and 8.0 hours. Since the heating process will be completed at some time in future, the **work** statement on line 65 places the process notice for the ingot back in the event list with an activation time equal to **TIME.V** plus heating time.

During the heating process, control passes to the timing routine, which determines the next event. Other ingots may also be processed while the process notice for this ingot is in the event list. When the process notice of this event is removed again from the event list (i.e. heating time expires), line 66—which simulates the removal of the ingot from the furnace—is executed. As line 68 is executed, a soaking pit is made available. Before the ingot process notice is removed from the system, the number of processed ingots is updated (line 70). When line 72 is executed, the process notice for the ingot is destroyed and control returns to the timing routine.

```
055  process INGOT
056
057     define ARRIVETIME as a real variable
058
059     let ARRIVETIME = TIME.V
060
061     request 1 PIT(1)
062     let WAIT.TIME = TIME.V - ARRIVETIME
063
064        file INGOT in FURNACE.SET
065        work UNIFORM.F(4.0, 8.0, 2) hours
066        remove INGOT from FURNACE.SET
067
068     relinquish 1 PIT(1)
069
070     add 1 to TOTAL.INGOTS
071
072  end
```

Figure 2-7. Listing for the Process Routine `INGOT`, Model I

Finally, the process routine for process **`STOP.SIM`** is given in figure 2-8. This process routine, as stated in **`main`**, is activated at the end of the simulation (when **`TIME.V = ENDTIME`**). It prints final statistics such as the total number of processed ingots, waiting time statistics and furnace utilization statistics. When the **`stop`** statement on line 98 is executed, it signals the completion of the simulation and returns control to the operating system.

```
process STOP.SIM

   print 6 lines with TIME.V, TOTAL.INGOTS thus
   Report After ****.** Simulated Hours - **** Ingots Processed


         -- All Times in Hours --


   print 5 lines with MEAN.WAIT.TIME, VAR.WAIT.TIME,
      MAX.WAIT.TIME thus
-- INGOT WAITING TIME STATISTICS
   MEAN WAIT TIME               ***.**
   VARIANCE                     ***.**
   MAXIMUM WAIT TIME            ***.**


   print 5 lines with MEAN.NO.OF.INGOTS, VAR.NO.OF.INGOTS,
      MAX.NO.OF.INGOTS, MIN.NO.OF.INGOTS thus
-- FURNACE UTILIZATION STATISTICS
   MEAN NO. OF INGOTS           **
   VARIANCE                     **.**
   MAXIMUM NO. OF INGOTS        **
   MINIMUM NO. OF INGOTS        **

   stop

end
```

Figure 2-8. Listing for the Process Routine `STOP.SIM`, Model I

2.3 Simulation Input and Output Analysis of Model I

Using the following parameters, the model produced the output shown in figure 2-9:

Number of soaking pits	7
Ingot interarrival times	Exponentially distributed with a mean of 1.5 hours
Ingot heating times	Uniformly distributed between 4 and 8 hours

Results indicate that very little waiting is involved in this case, although the furnace does reach its capacity of 7 ingots. The maximum wait time is 2.56 hours but the mean wait time is only 0.06 hours, with a variance of 0.05. We conclude that most of the 479 ingots processed are immediately transferred into the furnace with no waiting. The mean occupancy of the furnace is 4 ingots with a variance of 3.31. One would probably conclude that the furnace is underutilized in this situation.

```
   Report After  720.00 Simulated Hours - 479 Ingots Processed

                      -- All Times in Hours --

 --   INGOT WAITING TIME STATISTICS
      MEAN WAIT TIME                  .06
      VARIANCE                        .05
      MAXIMUM WAIT TIME              2.63

 --   FURNACE UTILIZATION STATISTICS
      MEAN NO. OF INGOTS             4
      VARIANCE                       3.31
      MAXIMUM NO. OF INGOTS          7
      MINIMUM NO. OF INGOTS          0
```

Figure 2-9. Simulation Output for Model I

2.4 Model II: Problem Statement

Model II introduces the combined continuous-discrete features of SIMSCRIPT II.5. It is model I with some modifications.

Let us assume that the change in ingot temperature is determined using the following differential equation:

$dh_i/dt = (H - h_i) * c_i$ equation 2.1

where:

- h_i is the temperature of the i^{th} ingot
- H is the furnace temperature; and
- c_i is the heating time coefficient of the i^{th} ingot and is equal to (0.07 + **x**), where **x** is normally distributed with a mean of 0.05 and a standard deviation of 0.01.

Ingots are heated toward a desired target temperature which is uniformly distributed in the interval 800 to 1000° F. Ingot initial temperatures are uniformly distributed in the interval from 100 to 200° F and if there is no soaking pit when an ingot arrives, the ingot will be stored in a warming pit where it preserves its initial temperature. When a soaking pit becomes available, the ingot is processed. Assuming that the furnace temperature is constant at 1500° F, simulate the heating process for 30 days (720 hours) and record the waiting time, the furnace utilization, and the final temperature distribution statistics.

2.5 Implementation of Model II

The furnace system in this model is slightly different from the previous example. While ingot arrivals are still discrete events, the heating time is no longer pre-determined. To determine the heating time of the ingots, the ingot temperatures are continuously evaluated using equation 2.1 until the ingots reach the desired final temperatures.

Implementation of the model in this form requires combined continuous-discrete capabilities. SIMSCRIPT will now be used to model this system.

Since the differential equations must be associated with a process notice, and since continuous variables may only be defined as attributes of processes, some changes have to be made to the definition of process **INGOT**. The process will be modified to include such attributes of ingots as current and final temperatures. The process **INGOT** definition could be:

```
every INGOT has
      a CURRENT.TEMP,
      a HEAT.COEFF,
      a FINAL.TEMP
and may belong to the FURNACE.SET
```

The attribute **CURRENT.TEMP** represents the current temperature of an ingot. It is continuously evaluated, and must therefore be declared as a continuous variable:

```
define CURRENT.TEMP as a continuous double variable
```

Since the ingot temperature is continuously evaluated until it reaches a desired final temperature, the work statement must also be modified to relate to this continuous variable:

```
work continuously evaluating 'HEATINGOT' testing 'HOTENOUGH'
```

where **HEATINGOT** is the routine which includes the differential equations to be continuously evaluated. The other routine, **HOTENOUGH**, signals when an ingot reaches its desired final temperature. Further explanation will be provided below. The block diagram in figure 2-10 illustrates the actions of the individual subprograms and their calling sequences.

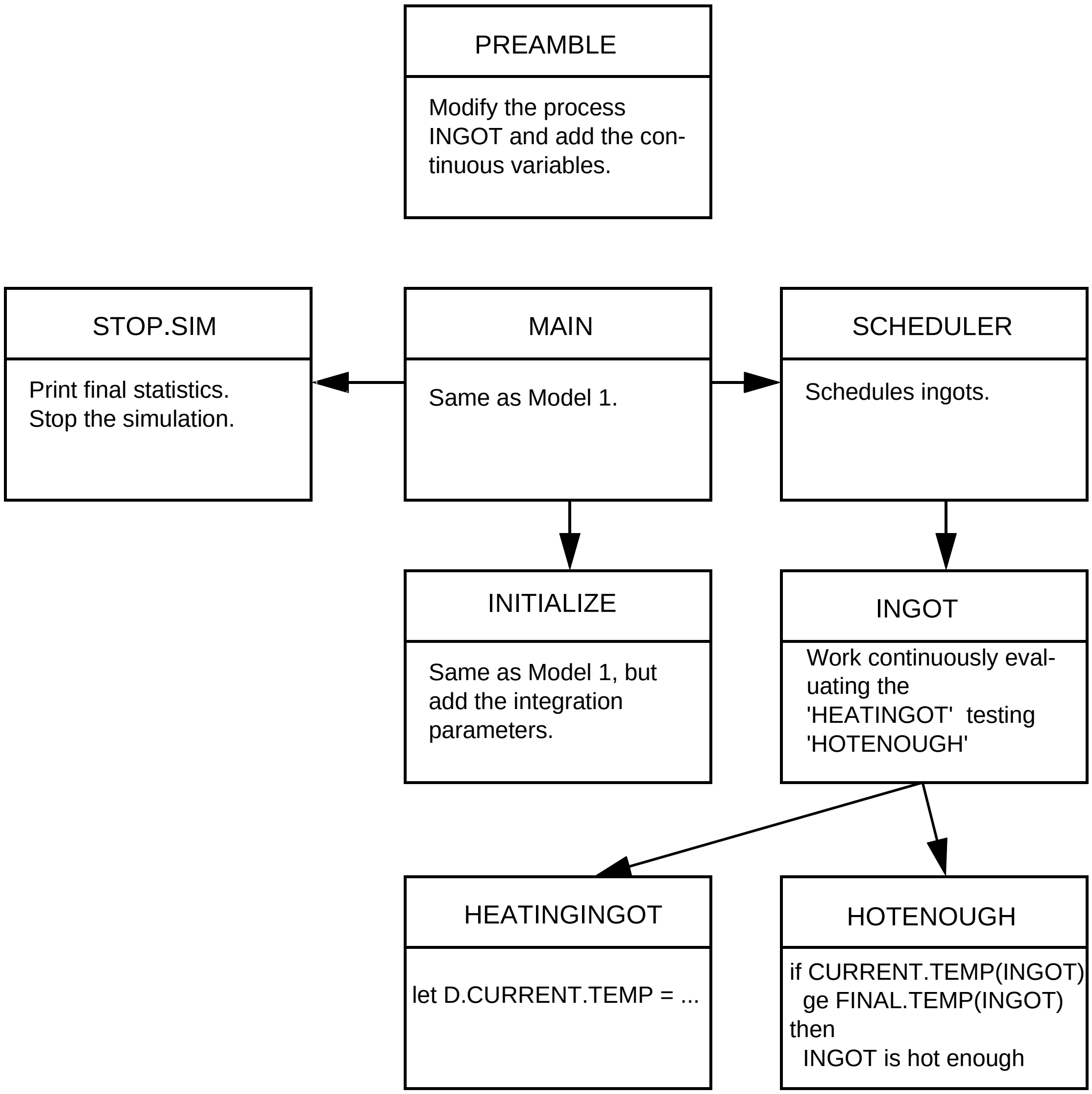

Figure 2-10. SIMSCRIPT Model II Segments

The **`preamble`** for this model is given in figure 2-11. Lines 13 through 15 define the type of process **`INGOT`** attributes. Line 13 defines the attribute **`current.temp`** as a continuous double variable. This is the variable which will continuously be evaluated. The types of the attributes **`FINAL.TEMP`** and **`HEAT.COEFF`** are declared on lines 14 and 15, respectively. Line 23 defines three variables, **`WAIT.TIME, HEAT.TIME, END.TIME`**. Two of these variables were defined in the first model; the third, **`HEAT.TIME`**, is used to hold information on ingot heating times. Line 24 introduces another new variable, **`LEAVE.TEMP`**, which is used to hold information on final ingot temperatures. This variable is used in the **`tally`** statement on line 41. The **`accumulate`** statement starting on line 31 computes statistics on the ingot heating times (**`mean, variance, maximum`**, and **`minimum`**). The **`tally`** statement on line 41 prepares a histogram of ingot final temperatures.

```
preamble

normally mode is undefined

   processes include STOP.SIM, SCHEDULER

      every INGOT has
         a CURRENT.TEMP,
         a HEAT.COEFF,
         a FINAL.TEMP
      and may belong to the FURNACE.SET

   define CURRENT.TEMP as a continuous double variable
   define FINAL.TEMP as a double variable
   define HEAT.COEFF as a real variable

   THE SYSTEM owns the FURNACE.SET

   resources include PIT

   define FURNACE.TEMP as a double variable
   define TOTAL.INGOTS as an integer variable
   define ENDTIME, WAIT.TIME, HEAT.TIME as real variables
   define LEAVE.TEMP as a double variable
   define HOURS to mean units  ''of time

   accumulate MEAN.WAIT.TIME as the mean,
              VAR.WAIT.TIME as the variance,
              MAX.WAIT.TIME as the maximum of WAIT.TIME

   accumulate MEAN.HEAT.TIME as the mean,
              VAR.HEAT.TIME as the variance,
              MAX.HEAT.TIME as the maximum,
              MIN.HEAT.TIME as the minimum of HEAT.TIME

   accumulate MEAN.NO.OF.INGOTS as the mean,
              VAR.NO.OF.INGOTS as the variance,
              MAX.NO.OF.INGOTS as the maximum,
              MIN.NO.OF.INGOTS as the minimum of N.FURNACE.SET

   tally TLEAVE(800.0 TO 1000.0 by 5)as the histogram of LEAVE.TEMP

   define HOTENOUGH as an integer function

end
```

Figure 2-11. Listing for the PREAMBLE, Model II

The **main** program of this model is given in figure 2-12. There are no changes from Model I.

```
main

   call INITIALIZE

   start simulation

end
```

Figure 2-12. Listing for the MAIN Program, Model II

The **INITIALIZE** routine appears in figure 2-13. Line 55 explicitly sets the integration routine to **RUNGE.KUTTA.R**. This is the default, and the statement could be removed (implicit definition). Lines 59 through 62 initialize the integration parameters **MAX.STEP.V**, **MIN.STEP.V**, **ABS.ERR.V**, and **REL.ERR.V**, which describe the maximum step size, minimum step size, absolute error tolerance, and relative error tolerance, respectively. The rest of **INITIALIZE** is the same as in the previous example.

```
053   routine INITIALIZE
054
055      let INTEGRATOR.V = 'RUNGE.KUTTA.R'
056      open 1 for input, name is "IN.DAT"
057      use 1 for input
058
059      read MAX.STEP.V
060      read MIN.STEP.V
061      read ABS.ERR.V
062      read REL.ERR.V
063
064      create every PIT(1)
065      read U.PIT(1)
066
067      read FURNACE.TEMP    '' furnace initial temperature
068
069      read ENDTIME
070
071      let MIN.HEAT.TIME = INF.C  '' initialize MIN.HEAT.TIME
072
073      activate a SCHEDULER now
074
075      activate a STOP.SIM in ENDTIME hours
076
077   end
```

Figure 2-13. Listing of the Initialize Routine, Model II

Figure 2-14 shows the process routine for the process **SCHEDULER**. Line 83 assigns each ingot an initial temperature which is uniformly distributed between 100 and 200° F. An explicit reference is made to the **INGOT** notice (**CURRENT.TEMP(INGOT)**). This style will be followed throughout the entire model.

```
078  process SCHEDULER
079
080     while TIME.V lt ENDTIME
081        do
082           create an INGOT
083           let CURRENT.TEMP(INGOT)=UNIFORM.F(100.0,200.0,2)
084           activate INGOT now
085           wait EXPONENTIAL.F(1.5, 1) hours
086        loop
087
088  end
```

Figure 2-14. Listing for the Process Routine SCHEDULER, Model II

Figure 2-15 gives the process routine for process **INGOT**. Line 94 assigns a random heating coefficient to the ingot being processed. On line 95 the ingot is assigned a target temperature to which the ingot is to be heated. Line 102 is the major change in this routine:

```
work continuously evaluating 'HEATINGOT' testing 'HOTENOUGH'
```

This **work** statement replaces the **work** statement on line 65 in Model I (**work UNIFORM.F (4.0, 8.0, 2) hours**) in which the heating time is randomly chosen from a uniform distribution with a mean of 6 and a standard deviation of 2. The new **work** statement on line 102 involves two subprograms, the routine **HEATINGOT** and the function **HOTENOUGH.** This statement is discussed further below.

```
089   process INGOT
090
091      define ARRIVETIME, STARTTIME as double variables
092
093      let ARRIVETIME = TIME.V
094      let HEAT.COEFF(INGOT) = NORMAL.F(0.05, 0.01, 3) + 0.07
095      let FINAL.TEMP(INGOT) = UNIFORM.F(800.00, 1000.0, 4)
096
097      request 1 PIT(1)
098         let WAIT.TIME = TIME.V - ARRIVETIME
099
100         file INGOT in FURNACE.SET
101         let STARTTIME = TIME.V
102         work continuously evaluating 'HEATINGOT' testing 'HOTENOUGH'
103         let HEAT.TIME = TIME.V - STARTTIME
104         let LEAVE.TEMP = CURRENT.TEMP(INGOT)
105         remove INGOT from FURNACE.SET
106
107      relinquish 1 PIT(1)
108
109      add 1 to TOTAL.INGOTS
110
111   end
```

Figure 2-15 Listing for the Process Routine INGOT, Model II

The **HEATINGOT** routine is defined in figure 2-16. Its single argument is the associated process notice pointer. The routine includes the derivatives associated with the ingot process notice. Line 115 which is:

```
let D.CURRENT.TEMP(INGOT) = .....
```

defines the rate of change in ingot temperature ($dh_i/dt = (H - h_i) * c_i$). When routine **HEATINGOT** is invoked in a **work** statement (for example, line 102 of the **INGOT** routine), numerical integration (in this example **RUNGE KUTTA**) repeatedly evaluates the changes in the ingot temperature. The routine is invoked several times during a single integration step.

```
112   routine HEATINGOT (INGOT)
113      define INGOT as a pointer variable
114
115      let D.CURRENT.TEMP(INGOT)
116         = (FURNACE.TEMP - CURRENT.TEMP(INGOT)) * HEAT.COEFF(INGOT)
117
118   end
```

Figure 2-16. Listing for the Routine HEATINGOT, Model II

Next is the function **HOTENOUGH**, shown in figure 2-17. This function, when invoked, tests the current temperature of an ingot. It returns zero if the ingot has not reached its target temperature in the previous time step; otherwise it returns 1. Satisfaction of the tested condition (line 122) means the termination of integration for that specific process notice. Again, remember that the condition-testing routine is called several times during an integration step.

```
119   function HOTENOUGH (INGOT)
120      define INGOT as a pointer variable
121
122      if CURRENT.TEMP(INGOT) ge FINAL.TEMP(INGOT)
123         return with 1
124      endif
125
126      return with 0
127
128   end
```

Figure 2-17. Listing for Function `HOTENOUGH`, Model II

Finally, the routine for process **`STOP.SIM`** prints the ingot statistics (figure 2-18).

```
process STOP.SIM

   print 6 lines with TIME.V, TOTAL.INGOTS thus
   Report After ****.** Simulated Hours - **** Ingots Processed


            -- All Times in Hours --


   print 5 lines with MEAN.WAIT.TIME, VAR.WAIT.TIME, MAX.WAIT.TIME thus
-- INGOT WAITING TIME STATISTICS
   MEAN WAIT TIME              ***.**
   VARIANCE                    ***.**
   MAXIMUM WAIT TIME           ***.**


   print 6 lines with MEAN.HEAT.TIME, VAR.HEAT.TIME,
      MAX.HEAT.TIME, MIN.HEAT.TIME thus
-- INGOT HEATING TIME STATISTICS
   MEAN HEATING TIME           ***.**
   VARIANCE                    ***.**
   MAXIMUM HEATING TIME        ***.**
   MINIMUM HEATING TIME        ***.**


   print 5 lines with MEAN.NO.OF.INGOTS, VAR.NO.OF.INGOTS,
      MAX.NO.OF.INGOTS, MIN.NO.OF.INGOTS thus
-- FURNACE UTILIZATION STATISTICS
   MEAN NO. OF INGOTS          **
   VARIANCE                    **.**
   MAXIMUM NO. OF INGOTS       **
   MINIMUM NO. OF INGOTS       **


   use 5 for input

   write as "HIT ENTER FOR HISTOGRAM OF FINAL TEMPERATURE..", /
   read as /

   write as *
   display histogram TLEAVE

   stop

end
```

Figure 2-18. Listing for the Process Routine STOP.SIM, Model II

2.6 Simulation Input and Output Analysis of Model II

Using the following parameters, the model produced the output in figures 2-19a and 2-19b.

Simulation period	30 days (720 hours)
Number of soaking pits	7
Ingot interarrival times	Exponentially distributed with a mean of 1.5 hours
Furnace temperature	Steady at 1500 ^{o}F
Maximum step size	0.01
Minimum step size	0.001
Absolute error	0.0005
Relative error	0.05

This model is not equivalent in any sense to Model I, so comparisons between the two sets of results are not informative. Note that the mean number of ingots in the furnace is 5 with a variance of 3.19, and the furnace does fill up at times. The maximum wait time is 3.15 hours, with a mean of 0.13 hours and a variance of 0.16 hours. This suggests that most ingots are moved into the furnace at or shortly after their arrival. The furnace is quite well utilized in this case. Additional statistics are presented on the heating times of the ingots from the initial temperatures (uniformly distributed between 100 and 200° F) to the final temperatures (uniformly distributed between 800 and 1000° F). These vary between a minimum of 4.68 and a maximum of 9.79 hours, with a mean 6.71 and a variance of 1.41. Figure 2-19b gives a histogram of the final ingot temperatures for this run.

```
Report After  720.00 Simulated Hours - 480 Ingots Processed

                  -- All Times in Hours --

-- INGOT WAITING TIME STATISTICS
   MEAN WAIT TIME              .13
   VARIANCE                    .16
   MAXIMUM WAIT TIME          3.15

-- INGOT HEATING TIME STATISTICS
   MEAN HEATING TIME          6.71
   VARIANCE                   1.41
   MAXIMUM HEATING TIME       9.79
   MINIMUM HEATING TIME       4.68

-- FURNACE UTILIZATION STATISTICS
   MEAN NO. OF INGOTS         5
   VARIANCE                   3.19
   MAXIMUM NO. OF INGOTS      7
   MINIMUM NO. OF INGOTS      0
```

Figure 2-19a. Simulation Output for Model II

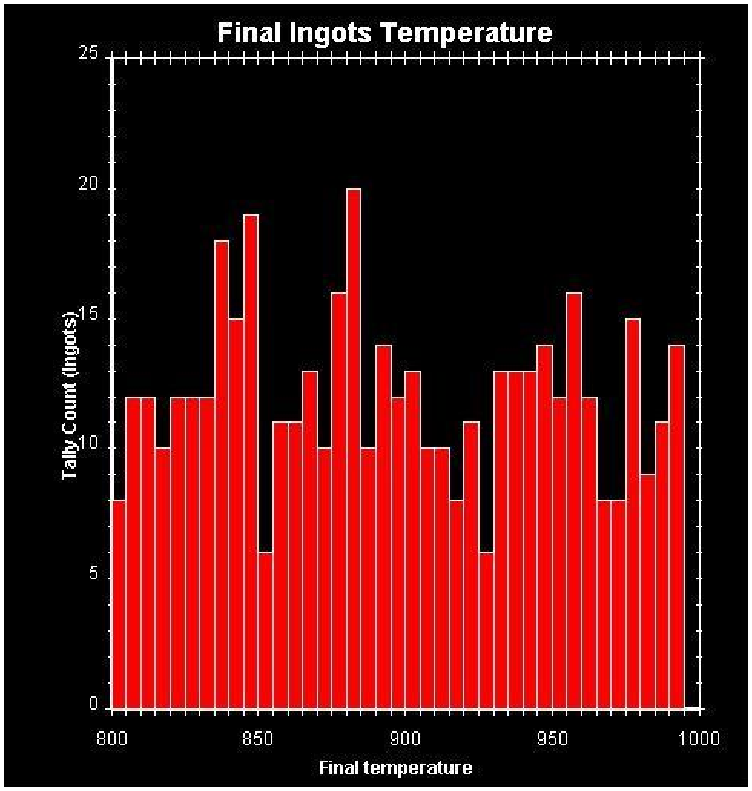

Figure 2-19b. Ingot Final Temperature Distribution, Model II

2.7 Model III: Problem Statement

In the two previous models the furnace temperature was assumed to be constant during the course of the simulation. This assumption was made for the sake of simplicity. In reality, the furnace temperature is normally increasing towards a certain final temperature. It also declines when cold ingots are put into the furnace. Introducing these changes to the model adds another level of complexity.

Assume that the furnace temperature, H, approaches a target temperature of 2500° F. The change in the furnace temperature is described by the following differential equation:

```
dH/dt = (2500 - H) * 0.05 equation 2.2
```

Normally H is less than 2500°, which means dH/dt is positive and temperature increases. As H approaches 2500° the rate of increase tends to zero.

2.8 Implementation of Model III

The change in furnace temperature is determined using equation 2.2, which must be continuously evaluated and updated as time passes. To model this in SIMSCRIPT, a continuous variable, **`FURNACE.TEMP`**, must be used. Since continuous variables are only defined as attributes of a process, a new process, **FURNACE**, must be introduced. This is defined as:

```
every FURNACE has
a FURNACE.TEMP
and owns a FURNACE.SET
define FURNACE.TEMP as a continuous double variable
```

This revises the processes definition previously introduced. In addition to this revision, two more routines are added to the previous model. First is the process routine associated with the process furnace. Second is the routine, **HEATUP**, which incorporates the differential equation associated with the furnace process. This routine includes equation 2.2 in the statement:

```
let D.FURNACE.TEMP(FURNACE) = (2500 - FURNACE.TEMP(FURNACE))*0.05
```

There is only one **FURNACE** process notice. This process notice is activated at time 0 and continues to update the furnace temperature throughout the entire course of simulation. This is accomplished by including following **work** statement in the furnace process routine:

```
work continuously evaluating 'HEATUP'
```

where **HEATUP** is the routine containing the differential equation describing the change in the furnace temperature. The block diagram in figure 2-20 illustrates the actions of the individual subprograms and their calling sequences.

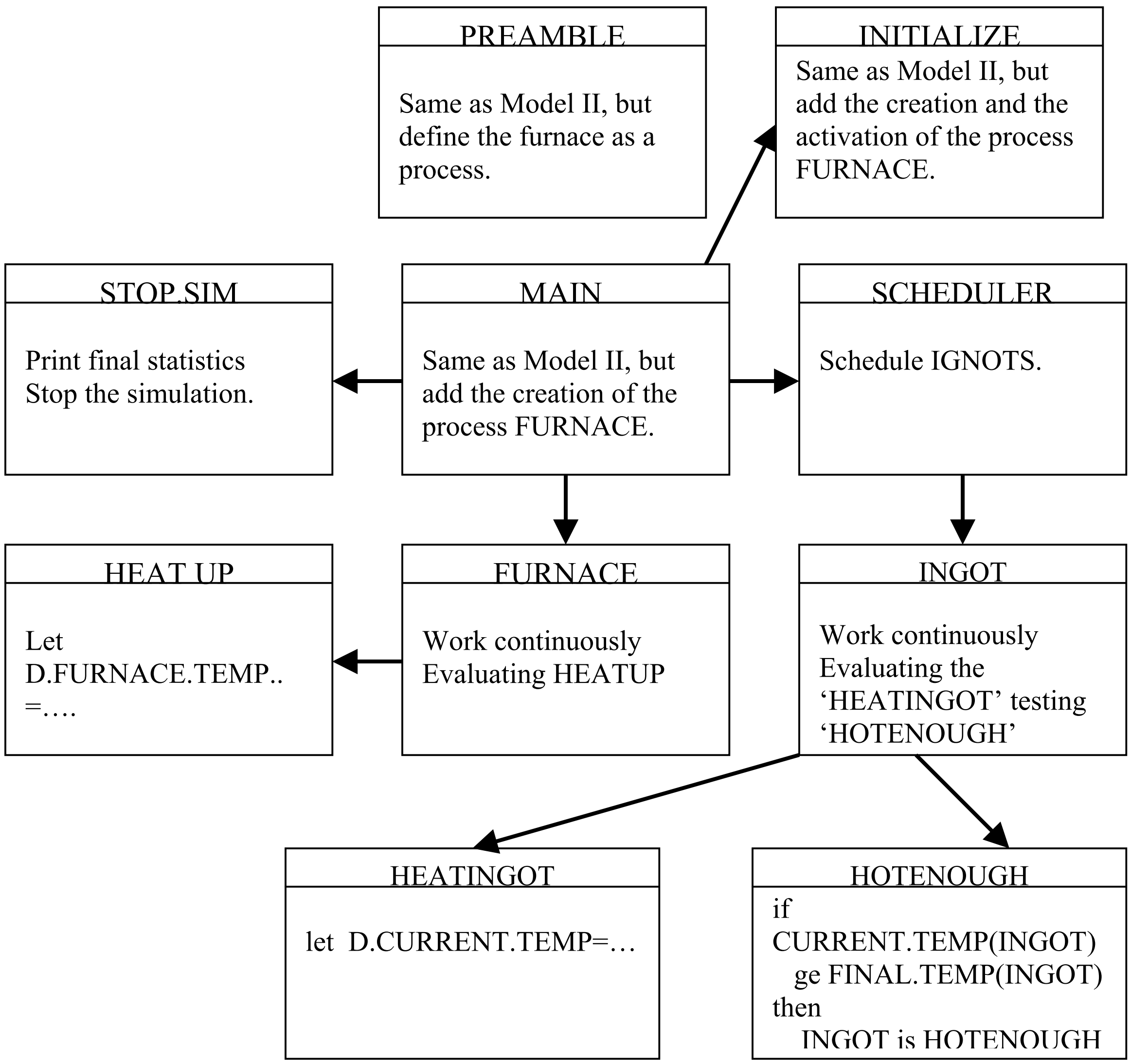

Figure 2-20. SIMSCRIPT Model III Segments

The **preamble** of this model is included in figure 2-21. It includes the changes discussed earlier, but otherwise resembles previous preambles. Lines 17 through 21 define the process **FURNACE** with one attribute, **FURNACE.TEMP**. The process owns the **FURNACE.SET**, which was previously owned by the **system**. Line 21 declares **FURNACE.TEMP** as a continuous double variable. No other changes are made.

```
preamble

   normally mode is undefined

   processes include STOP.SIM, SCHEDULER

      every INGOT has
         a CURRENT.TEMP,
         a FINAL.TEMP,
         a HEAT.COEFF
      and may belong to the FURNACE.SET

   define CURRENT.TEMP as a continuous double variable
   define FINAL.TEMP as a double variable
   define HEAT.COEFF as a real variable

      every FURNACE has
         a FURNACE.TEMP
      and owns a FURNACE.SET

   define FURNACE.TEMP as a continuous double variable

   resources include PIT

   define TOTAL.INGOTS as an integer variable
   define ENDTIME, WAIT.TIME, HEAT.TIME as real variables
   define LEAVE.TEMP as a double variable
   define hours to mean units  ''of time

   accumulate    MEAN.WAIT.TIME     as the mean,
                 VAR.WAIT.TIME      as the variance,
                 MAX.WAIT.TIME      as the maximum of WAIT.TIME

   accumulate    MEAN.HEAT.TIME     as the mean,
                 VAR.HEAT.TIME      as the variance,
                 MAX.HEAT.TIME      as the maximum,
                 MIN.HEAT.TIME      as the minimum of HEAT.TIME

   accumulate    MEAN.NO.OF.INGOTS  as the mean,
                 VAR.NO.OF.INGOTS   as the variance,
                 MAX.NO.OF.INGOTS   as the maximum,
                 MIN.NO.OF.INGOTS   as the minimum of N.FURNACE.SET

   tally TLEAVE(800.0 TO 1000.0 by 5)     as the histogram of LEAVE.TEMP

   define HOTENOUGH as an integer function

end
```

Figure 2-21. Listing for the PREAMBLE, Model III

The **main** program of this model appears in figure 2-22. There are no changes from the previous model.

```
main

   call INITIALIZE

   start simulation

end
```

Figure 2-22. Listing for the MAIN program, Model III

The **initialize** routine appears in figure 2-23. The only change to this routine is the addition of the creation of process **FURNACE**, lines 70 through 72. Line 70 creates process **FURNACE** and line 71 assigns it an initial temperature. Line 72 schedules the activation of the **FURNACE** process at the current value of **TIME.V** (currently equal to zero). No other changes are made.

```
056   routine INITIALIZE
057
058      let INTEGRATOR.V = 'RUNGE.KUTTA.R'
059      open 1 for input, name is "IN.DAT"
060      use 1 for input
061
062      read MAX.STEP.V
063      read MIN.STEP.V
064      read ABS.ERR.V
065      read REL.ERR.V
066
067      create every PIT(1)
068      read U.PIT(1)
069
070      create a FURNACE
071      read FURNACE.TEMP(FURNACE) '' Furnace Initial Temperature
072      activate FURNACE now
073
074      read ENDTIME
075
076      activate a STOP.SIM in ENDTIME hours
077
078      activate a SCHEDULER now
079
080      let MIN.HEAT.TIME = INF.C  '' initialize MIN.HEAT.TIME
081
082   end
```

Figure 2-23. Listing for the Initialize Routine, Model III

The process routine for the **SCHEDULER** process is given in figure 2-24. No changes are made.

```
process SCHEDULER

   while TIME.V lt ENDTIME
      do
         create an INGOT
         let CURRENT.TEMP(INGOT)=UNIFORM.F(100.0,200.0,2)
         activate INGOT now
         wait EXPONENTIAL.F(1.5, 1) hours
      loop

end
```

Figure 2-24. Listing for the Process Routine SCHEDULER, Model III

The process routine for the **INGOT** process appears in figure 2-25. The only change to this routine is the inclusion of the effect of adding cold ingots to the furnace. This is modeled on lines 105 through 109.

```
094   process INGOT
095
096      define ARRIVETIME, STARTTIME as double variables
097
098      let ARRIVETIME = TIME.V
099      let HEAT.COEFF(INGOT) = NORMAL.F(0.05, 0.01, 3) + 0.07
100      let FINAL.TEMP(INGOT) =UNIFORM.F(800.00, 1000.0, 4)
101
102      request 1 PIT(1)
103         let WAIT.TIME = TIME.V - ARRIVETIME
104
105
106
107
108
109
110
111         file INGOT in FURNACE.SET
112         let STARTTIME = TIME.V
113         work continuously evaluating 'HEATINGOT' testing 'HOTENOUGH'
114         let HEAT.TIME = TIME.V - STARTTIME
115         let LEAVE.TEMP = CURRENT.TEMP(INGOT)
116         remove INGOT from FURNACE.SET
117
118      relinquish 1 PIT(1)
119
120      add 1 to TOTAL.INGOTS
121
122   end
```

Figure 2-25. Listing for the Process Routine INGOT, Model III

The **HEATINGOT** routine is given in figure 2-26. The only change to this routine is the reference to the furnace temperature. In the previous model, the furnace temperature was considered to be constant. In this model the furnace temperature is referenced as an attribute of the process **FURNACE** (**FURNACE.TEMP(FURNACE**)). Remember, there is only one process **FURNACE** in the model.

```
routine HEATINGOT (INGOT)
   define INGOT as a pointer variable

   let D.CURRENT.TEMP(INGOT)
      = (FURNACE.TEMP(FURNACE) - CURRENT.TEMP(INGOT)) * HEAT.COEFF(INGOT)

end
```

Figure 2-26. Listing for the Routine HEATINGOT, Model III

Function **HOTENOUGH** is given in figure 2-27. It is unchanged.

```
function HOTENOUGH (INGOT)
   define INGOT as a pointer variable

   if CURRENT.TEMP(INGOT) ge FINAL.TEMP(INGOT)
      return with 1
   endif

   return with 0

end
```

Figure 2.27 Listing for Function HOTENOUGH, Model III

The process routine for process **FURNACE** is given in figure 2-28. Only one **FURNACE** process occurs in the model. This process model is activated at the start of the simulation (i.e., at time 0). The only code associated with process **FURNACE** is the statement:

```
work continuously evaluating 'HEATUP'
```

where **HEATUP** is the routine containing the differential equation associated with the process notice. As a result of invoking the **HEATUP** routine in the **work continuously** statement, numerical integration (in this example **RUNGE KUTTA**) is used repeatedly to evaluate changes in furnace temperature.

```
process FURNACE

   work continuously evaluating 'HEATUP'

end
```

Figure 2-28. Listing for the Process Routine FURNACE, Model III

In figure 2-29 the **HEATUP** routine describes the differential equations associated with the **FURNACE** process notice. Its single argument is the **FURNACE** process notice pointer. Line 148 computes the change in the furnace temperature which is defined by:

```
dH/dt = (2500 - H) * 0.05
```

```
145   routine HEATUP (FURNACE)
146      define FURNACE as a pointer variable
147
148       let D.FURNACE.TEMP(FURNACE)=2500-FURNACE.TEMP(FURNACE))*0.05
149
150   end
```

Figure 2-29. Listing for the Routine HEATUP, Model III

Finally, the process routine for process **STOP.SIM** is unchanged. See figure 2-30.

```
process STOP.SIM

   print 6 lines with TIME.V, TOTAL.INGOTS thus
   Report After ****.** Simulated Hours - **** Ingots Processed


            -- All Times in Hours --


   print 5 lines with MEAN.WAIT.TIME, VAR.WAIT.TIME, MAX.WAIT.TIME THUS
-- INGOT WAITING TIME STATISTICS
   MEAN WAIT TIME             ***.**
   VARIANCE                   ***.**
   MAXIMUM WAIT TIME          ***.**


   print 6 lines with MEAN.HEAT.TIME, VAR.HEAT.TIME, MAX.HEAT.TIME,
   MIN.HEAT.TIME thus
-- INGOT HEATING TIME STATISTICS
   MEAN HEATING TIME          ***.**
   VARIANCE                   ***.**
   MAXIMUM HEATING TIME       ***.**
   MINIMUM HEATING TIME***.**


   print 5 lines with MEAN.NO.OF.INGOTS, VAR.NO.OF.INGOTS,
      MAX.NO.OF.INGOTS, MIN.NO.OF.INGOTS thus
-- FURNACE UTILIZATION STATISTICS
   MEAN NO. OF INGOTS         **
   VARIANCE                   **.**
   MAXIMUM NO. OF INGOTS      **
   MINIMUM NO. OF INGOTS      **

   Use 5 for input

   write as "HIT ENTER FOR HISTOGRAM OF FINAL TEMPERATURE..", /
   read as /

   write as *
   display histogram TLEAVE

   stop

end
```

Figure 2-30. Listing for the Process Routine `STOP.SIM`, Model III

2.9 Simulation Input and Output Analysis of Model III

Using the following parameters, the model produced the output given in figures 2-31a and 2-31b.

Simulation period	30 days (720 hours)
Number of soaking pits	7
Ingot interarrival times	Exponentially distributed with a mean of 1.5 hours
Furnace initial temperature	1000° F
Furnacemaximum temperature	2500° F
Maximum step size	0.01
Minimum step size	0.001
Absolute error	0.0005
Relative error	0.05

The results in figure 2-31a show the effect of the higher average furnace temperature. Although the furnace does fill up, the mean utilization is only 2 ingots. There is some waiting (maximum wait time is 1.07), but the mean wait time is reported as zero, implying an average value of less than 0.005 hours. The ingot heating times are also reduced, ranging from 2.25 to 9.06 hours, with a mean of 3.30 hours and a variance of 0.50 hours. The furnace is badly underutilized. Figure 2-31b gives a histogram of the final ingot temperatures.

```
Report After  720.00 Simulated Hours - 482 Ingots Processed

      -- All Times in Hours --

-- INGOT WAITING TIME STATISTICS
   MEAN WAIT TIME                       .00
   VARIANCE                             .00
   MAXIMUM WAIT TIME                   1.07

-- INGOT HEATING TIME STATISTICS
   MEAN HEATING TIME                   3.30
   VARIANCE                             .50
   MAXIMUM HEATING TIME                9.06
   MINIMUM HEATING TIME                2.27

-- FURNACE UTILIZATION STATISTICS
   MEAN NO. OF INGOTS                  2
   VARIANCE                            2.31
   MAXIMUM NO. OF INGOTS               7
   MINIMUM NO. OF INGOTS               0
```

Figure 2-31a. Simulation Output for Model III

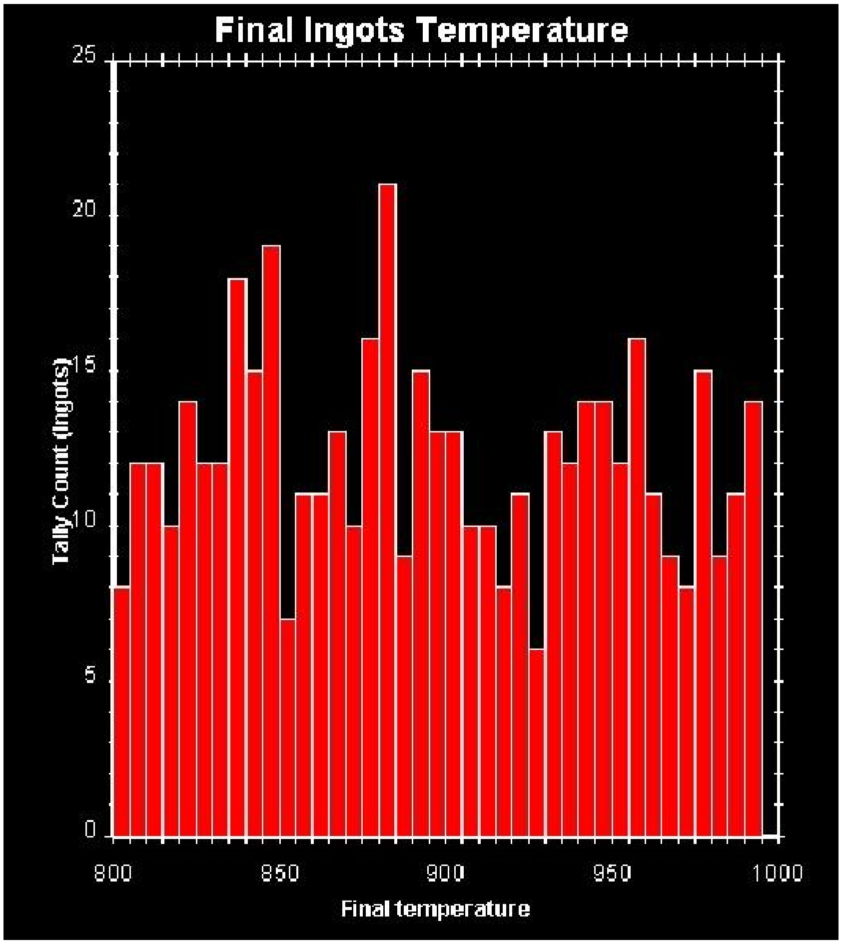

Figure 2-31b. Ingot Final Temperature Distribution, Model III

2.10 Suggested Exercises

1. Modify Model III so that the rate of change of furnace temperature is given by:

```
dH/dt = (2500 - H) * k
```

where k = 0.1 if there are less than 5 ingots in the furnace, and k = 0.05 for 5 or more ingots.

2. Modify Model II so that the furnace heating process stops if there are no ingots in the furnace. When this condition occurs, the furnace temperature is maintained at its current value. When an ingot arrives, the furnace heating process starts again.

3. Modify Model III to allow furnace maintenance to take place. The maintenance procedure is as follows. No new ingots are to be put into the furnace after 15 days (360 hours) until maintenance is complete. Process the ingots currently in the furnace and when the furnace is empty turn off the heaters. The furnace temperature is now defined by:

```
dH/dt = -0.1 * H
```

Wait until the furnace has cooled to 100° F; then hold the temperature constant for 4 hours while maintenance is performed. The furnace is then reheated (using equation 2.2). Recommence loading ingots into the furnace when its temperature reaches 500° F.

4. Partially-filled cylinders of pressurized gas are topped up at one of three available filling stations. Each filling station has a line pressure **`PL`** and each cylinder has a maximum pressure **`PM`**. The line pressures for the three stations are 750, 1000 and 1250 psi, with a mean of 500 psi and a standard deviation of 20 psi. The initial pressures in the cylinders are uniformly distributed in the range of 0 to 200 psi. When a cylinder is being filled its pressure, **`p`**, increases at a rate:

```
dP/dt = (PL-P) * a (measured in minutes)
```

where **`a`** is normally distributed with a mean of 0.5 and a standard deviation of 0.05. When **`P`** is equal to **`PM`**, filling of the cylinder stops. The arrival of cylinders is exponentially distributed with a mean of 2 minutes. Arriving cylinders can be assigned to any available filling station. If all stations are busy, cylinders wait in a single queue. You can collect and display statistics on various variables of the system.

REFERENCES

1. Golden, D. G., and J. D. Schoeffler: "*GSL - A Combined Continuous and Discrete Simulation Language*," SIMULATION, pp. 1-8, January 1973.

2. "*Ingots*," A Tutorial Model, Tutorial Diskette - PC Simlab, CACI Inc., Los Angeles, 1987.

BIBLIOGRAPHY

"*SIMSCRIPT II.5 Reference Handbook*," CACI.

Fahrland, D.: "*Combined Discrete Event Continuous Systems Simulation*," SIMULATION, pp. 61-72, February 1970.

Speckhart, F. H. and W. L. Green: "*A Guide to Using CSMP- THE Continuous System Modeling Program*," Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976.

"*SIMSCRIPT II.5 Programming Language*," CACI.

Law, A. M. and W. D. Kelton: " *Simulation Modeling and Analysis*," McGraw Hill, New York, 1982.

Payne, J. A.: "*INTRODUCTION TO SIMULATION Programming Techniques and Methods of Analysis*," McGraw-Hill, New York, 1982.

"*Building Simulation Models with SIMSCRIPT II.5*," CACI.

"*CSSL-IV*," Version Four, Reference Manual, Simulation Services, 20926 Germain Street, Chatsworth, California, 1984.

"*ISIM Simulation*," Fourth Edition, Reference Manual, CHA, P.O. Box 943, Chico, California, 95927, 1986.

APPENDIX A. Model III Listing

```
preamble

   normally mode is undefined

   processes include STOP.SIM, SCHEDULER

      every INGOT has
      a CURRENT.TEMP,
      a FINAL.TEMP,
      a HEAT.COEFF
   and may belong to the FURNACE.SET

   define CURRENT.TEMP as a continuous double variable
   define FINAL.TEMP as a double variable
   define HEAT.COEFF as a real variable

      every FURNACE has
         a FURNACE.TEMP
      and owns a FURNACE.SET

   define FURNACE.TEMP as a continuous double variable

   resources include PIT

   define TOTAL.INGOTS as an integer variable
   define ENDTIME, WAIT.TIME, HEAT.TIME as real variables
   define LEAVE.TEMP as a double variable
   define HOURS to mean units  ''of time

   accumulate   MEAN.WAIT.TIME  as the mean,
                VAR.WAIT.TIME   as the variance,
                MAX.WAIT.TIME   as the maximum of WAIT.TIME

   accumulate   MEAN.HEAT.TIME  as the mean.
                VAR.HEAT.TIME   as the variance,
                MAX.HEAT.TIME   as the maximum,
                MIN.HEAT.TIME   as the minimum of HEAT.TIME

   accumulate   MEAN.NO.OF.INGOTS as the mean,
                VAR.NO.OF.INGOTS as the variance,
                MAX.NO.OF.INGOTS as the maximum,
                MIN.NO.OF.INGOTS  as the minimum of N.FURNACE.SET

   tally TLEAVE(800.0 TO 1000.0 by 5) as the histogram of LEAVE.TEMP

   define HOTENOUGH as an integer function

end
```

```
main

   call INITIALIZE

   start simulation

end

routine INITIALIZE

   let INTEGRATOR.V = 'RUNGE.KUTTA.R'
   open 1 for input, name is "IN.DAT"
   use 1 for input

   read MAX.STEP.V
   read MIN.STEP.V
   read ABS.ERR.V
   read REL.ERR.V

   create every PIT(1)
   read U.PIT(1)

   create a FURNACE
   read FURNACE.TEMP(FURNACE)  '' furnace initial temperature
   activate FURNACE now

   read ENDTIME

   activate a STOP.SIM in ENDTIME hours

   activate a SCHEDULER now

   let MIN.HEAT.TIME = INF.C  '' initialize MIN.HEAT.TIME

end

process SCHEDULER

   while TIME.V lt ENDTIME
      do
         create an INGOT
         let CURRENT.TEMP(INGOT) = UNIFORM.F(100.0, 200.0, 2)
         activate INGOT now
         wait EXPONENTIAL.F(1.5, 1) hours
      loop

end
```

```
process INGOT

   define ARRIVETIME, STARTTIME as double variable

   let ARRIVETIME = TIME.V
   let HEAT.COEFF(INGOT) = NORMAL.F(0.05, 0.01, 3) + 0.07
   let FINAL.TEMP(INGOT) =UNIFORM.F(800.00, 1000.0, 4)

   request 1 PIT(1)
      let WAIT.TIME = TIME.V - ARRIVETIME







      file INGOT in FURNACE.SET
      let STARTTIME = TIME.V
      work continuously evaluating 'HEATINGOT' testing 'HOTENOUGH'
      let HEAT.TIME = TIME.V - STARTTIME
      let LEAVE.TEMP = CURRENT.TEMP(INGOT)
      remove INGOt from FURNACE.SET

   relinquish 1 PIT(1)

   add 1 to TOTAL.INGOTS

end
```

```
routine HEATINGOT (INGOT)
   define INGOT as a pointer variable

   let D.CURRENT.TEMP(INGOT)
      = (FURNACE.TEMP(FURNACE) - CURRENT.TEMP(INGOT)) * HEAT.COEFF(INGOT)

end
```

```
function HOTENOUGH (INGOT)
   define INGOT as a pointer variable

   if CURRENT.TEMP(INGOT) ge FINAL.TEMP(INGOT)
      return with 1
   endif

   return with 0

end
```

```
process FURNACE

   work continuously evaluating 'HEATUP'

end


routine HEATUP (FURNACE)
   define FURNACE as a pointer variable

   let D.FURNACE.TEMP(FURNACE) = (2500 - FURNACE.TEMP(FURNACE)) * 0.05

end


process STOP.SIM

   print 6 lines with TIME.V, TOTAL.INGOTS thus
   REPORT AFTER ****.** SIMULATED HOURS - **** INGOTS PROCESSED


             -- ALL TIMES IN HOURS --


   print 5 lines with MEAN.WAIT.TIME, VAR.WAIT.TIME, MAX.WAIT.TIME thus
-- INGOT WAITING TIME STATISTICS
   MEAN WAIT TIME              ***.**
   VARIANCE                    ***.**
   MAXIMUM WAIT TIME           ***.**


   print 6 lines with MEAN.HEAT.TIME, VAR.HEAT.TIME, MAX.HEAT.TIME,
      MIN.HEAT.TIME thus
-- INGOT HEATING TIME STATISTICS
   MEAN HEATING TIME           ***.**
   VARIANCE                    ***.**
   MAXIMUM HEATING TIME        ***.**
   MINIMUM HEATING TIME        ***.**


   print 5 lines with MEAN.NO.OF.INGOTS, VAR.NO.OF.INGOTS,
   MAX.NO.OF.INGOTS, MIN.NO.OF.INGOTS thus
-- FURNACE UTILIZATION STATISTICS
   MEAN NO. OF INGOTS                  **
   VARIANCE                            **.**
   MAXIMUM NO. OF INGOTS               **
   MINIMUM NO. OF INGOTS               **

   use 5 for input

```

```
    write as "HIT ENTER FOR HISTOGRAM OF FINAL TEMPERATURE..", /
    read as /

    write as *
    display histogram TLEAVE

    stop

end
```