JKAU: Eng. Sci., **vol. 10 no. 1**, pp. 27-41 (1418 A.H. / 1998 A.D.)

Cell Growth Simulation (CGS): The Simulator

OSMAN S. OZKUL[1*], SERAJ Y. ABED[1], MUSTAFA M. AL-IDRISI[1], SUFIAN EL-ASSOULI[2] and MAJID AMER[3]

[1]Department of Industrial Engineering, Faculty of Engineering,
[2]Department of Medical Biology, Faculty of Medicine,
King Abdulaziz University, Jeddah; and
[3]Department of Oncology, King Faisal Speciality Hospital and Research Center, Riyadh, Saudi Arabia

ABSTRACT. Cell Growth Simulation (CGS) is a software package that simulates the growth of cultured cells, and the effects of different types of agents on the cell populations in vitro. It is a stochastic simulation system and is based on the cell cycle kinetics. Experiments involving DNA synthesis, blocking, mitosis inhibition, labeling and cell kill can be simulated by imitating application of drugs such as thymidine, aphidicoline, hydroxyurea, vincristine, finblistine, colcemid and others. The model to be simulated is described by the user according to his theoretical convictions. The system provides numerous facilities to help the user describe an experiment and examine the simulation results. A menu driven interactive scheme is used to communicate with the user. CGS is implemented on AT type personal computers with hard disks.

This paper focuses on the simulator modules of CGS. It is an event driven simulator which follows the individual cells through their life cycles. General structure of the simulator, primary events and their attributes, chain operations, culture initializations, imitation of cell maturation and DNA synthesis are explained in the text as well as collection of statistics from the simulated experiments.

1. Introduction

Digital simulation is being used in cell biology since 1960's[1-5]. The computer programs for these studies and many others were written in procedure oriented languages such as Fortran, Algol and PL-I. Simulation languages such as Simscript, GPSS,

*To whom all correspondence should be addressed.

SLAM, etc., have rarely been used in simulating experiments in cell biology. Developing custom made computer programs for each study individually is costly and requires close cooperation between biologists, programmers and simulation people. Most of the time, this combination is difficult to achieve. In order to get around this problem, specialized simulation packages have been developed for cellular experiments[6-8], of which CELLSIM is the most widely known language.

Developments in computer technology have been very rapid in recent years and the beneficiaries are the users. A noncomputer professional can afford to put a powerful PC into his office and learn to use it without the help of a computer programmer. Cell Growth Simulation (CGS) is a software package that attempts to make digital simulation a practical research tool for cell biologists. Compared to its predecessors, CGS requires less learning time from the users and less effort during its use. It offers more capability and flexibility in modeling cellular experiments and examining its results. CGS is implemented on IBM compatible AT computers with hard disks and 512k byte RAM memory by using the True Basic language. It contains 8 logical modules, 5 physical modules, 35 start-up files, 209 subprograms, and about 6200 source lines. CGS is documented in three volumes: (I) Research and development behind it, (II) Systems manual for maintenance and update, (III) User manual for the biologists, totaling over 500 pages.

The overall design philosophy and general structure of CGS is presented in Ref. [9], its interaction with the user in a friendly manner is explained in Ref. [10]. This paper is directed toward the simulation modules of the package. The basic principles of simulation models imbedded into the design, assumptions, flexibilities and capabilities provided by the system, initialization problems of the cell cultures, execution logic of the event driven simulator, primary events and their attributes, double linked event-chain and its operations, imitation of the behavior of the cells through their life cycles and the influence of the drugs that disturb or stop the normal cell maturation, and simulation of DNA synthesis are explained in the upcoming sections. The paper ends with a discussion on collecting periodic information about the status of the experiment being simulated.

2. Basic Principles

Cell cultures are simulated by following the individual cells through their life cycles. Life cycle contains several phases and cells spend random amounts of time in each phase. When a cell splits into two daughter cells the life cycle starts all over again. The user defines the cell cycle and its parameters. This provides flexibility to the system and hopefully meets the needs of a larger user group and greater variety of cells. The system imposes very few restrictions on the life cycle, such as every cycle must have a DNA synthesis phase and a mitosis phase. Every phase must have at least one inlet and no more than three outlets, and the number of phases in the cycle must not exceed 19.

Cell population sizes run into millions. For reasons of execution efficiency and primary storage restrictions such large number of cells can not be followed by the simula tor. The simulator places the cells into 1000 typical cell groups and follows them through the cycle and keeps track of the number of cells in each group.

It is an event driven, variable time increment type simulator. Therefore, real time values do not pose any problems, the time units need not be integerized and the smallest time increment need not be found. Everything that changes the status of the model is considered an event and all the events are kept in a chronologically ordered double-linked chain. Execution sequence conflicts among the simultaneously occurring events are broken by using priority levels.

Effect of drugs on the cultured cells in vitro are imitated through a set of abstract facilities. The cells can be blocked, labeled or frozen, their development can be delayed or they can be killed. The simulator provides several options to the users for inoculum, and it can also take the final status of a culture for a previously run experiment as the initial condition of a new run. Formation of the new DNA is based on the time that the cell has spent in the synthesis phase, *i.e,* no external variation factors are used.

3. System Initialization

Most of the information that the simulator needs about the experiment are kept in four arrays. Some portions of these arrays are present by the experiment description and verification modules. Simulation preparation activities complete the initialization for these arrays. Phase related information are kept in two two-dimensional arrays holding integer and real type data respectively. Information such as phase specialization, transit time distribution, etc., are preset by experiment description and verification modules. The future events chain is initialized either by copying the contents from the disk for continued experiments, or by setting all the elements to zero and the starting location to one for new experiments.

4. Culture Initialization

Inoculum can be specified in a number of different ways by the user. They are briefly explained below.

a. Final Conditions of a Previous Simulation

If the user informs the system, CGS saves the ending conditions of the current run. This data can be used for initializing the inoculum of another experiment.

b. Asynchronous Growing Culture

There are approximately twice as many young cells in a growing culture as old cells and the decreases follows a logarithmic curve[11]. The experimental results confirm this theoretical curve closely in the middle of the cell cycle and deviate slightly at the end points. The system uses end points which are in between the theoretical and experimental points and approximates the middle by a straight line. The number of cells in the inoculum are randomly assigned to this trapezoid where each phase has a slice.

c. Steady State Culture

It is assumed that there are as many cell deaths as cell births in this option. The number of cells in the phases of cell cycle are proportional to their phase transit times. The system assigns the number of cells in inoculum randomly to a rectangle where each phrase has a slice.

d. Synchronous Cultures

The users can start the experiments with synchronized cultures to save time and cut down on the output. The system asks about the number of cells in each phase under this option and requires that at least one phase must have cells in it. The distribution of the cells within a phase needs to be known also. Under the default option, a trapezoid similar to asynchronous culture is assimilated for the phase and the cells are assigned to it randomly. The cells can also be spread around the phase evenly with uniform age differences, or they can all be placed to the beginning (zero age) or the end of the phase (full page).

5. Events, Attributes and Chain Operations

CGS contains an event driven simulator, therefore almost everything that is affecting the status of the model is treated as an event. Cell movements from one phase of the cycle to another are the most common events and has the highest priority. The attributes of this type of event are the event occurrence time, the number of cells in the group, label indicator, DNA amount, entry time to the phase, time to be spent in the phase and the priority level.

The other events in the system are simulation interrupts, collection of periodic statistics, simulation screen updates, end of simulation, cell kill, freeze, delay, label and block commands. The command particulars such as ending time, effectiveness rate, etc., are carried as attributes. Priority level is an attribute for all events and the end of simulation has the lowest priority.

The future event chain keeps the events in a chronological order. Four one-dimensional arrays contain the event codes, occurrence times, forward pointers and the backward pointers respectively. The attributes of the events are stored into a fifth array (a character array) in a bit-packed form in order to save memory. If there are more than one event happening at the same time, these events and their attributes are transferred into the current events table. This table is arranged according to the priority levels of the events before execution. Since cell movements has the highest priority they are never entered into this table.

The future events chain is operated mainly by two routines. One of them removes the next imminent event from the top of the chain, updates the pointers, return the event code, the event time and row reference number for the attributes. The other subroutine schedules a given event. It finds an empty row, locates the proper chronological spot in the chain, and places the event there.

6. Simulation Driver

The structure of the driver routine for the simulator is shown on Fig. 1. It controls and coordinates all the activities of this module. The details of its different segments are shown on consecutive figures. Basically it has three portions; preliminary activities, simulation within an endless loop and the ending activities. The figures contain a few names in parenthesis. They are the actual module names for some of the important functions.

```
begin simulation driver routine (m1)
    preparation activities
    simulation within an endless loop
    wrap-up activities
end routine
```

FIG. 1. Driver routine for the simulator.

The preparation activities are expanded in Fig. 2. When a simulation is requested by the user, the system asks, receives and checks the experiment id number, brings the experiment description into primary memory from the disk and opens the temporary results files to be used during simulation.

```
receive and check experiment id number
bring experiment into primary memory
open temporary result files
if this is a continued experiment then
    bring-in the final conditions of the previous run
    do selective initialization
    take the old external commands out of the chain
    schedule end of simulation
    collect statistics and schedule stat. collection
    schedule new commands
    setup simulation screen display
else
    call simulation preparation module for full
        system and inoculum initializations (k4)
end if
```

FIG. 2. Preliminary activities of the simulator.

The system and culture initializations differ depending on the status of the simulation. If it is a continuation of a previous run, the ending conditions of the culture for the earlier simulation are brought in from a hard disk and used as the inoculum for this run. Otherwise, simulation preparation module makes the necessary initializations. Initial statistics reflecting the current status of the culture is collected and the simulation screen is set up and displayed.

The core of the simulator is the endless loop in the middle. Its expansion is shown on Fig. 3. Although the figure is a big long, it has the following simple structure.

1. Fetch the next event(s)
2. Execute the event(s)
3. Go back to step (1) if end of simulation is not reached.

```
do
    find the next event and define its code, time
      and row reference number for attributes
      (cdr, tmr, rowr)
    select on event code (cdr)
    no more events (0)
      if the current events  table is empty then
        print error message
        set the flags to stop simulation
        exit do
      else
        execute current events (m3)
        if end of simulation, set flags and exit do
      end if
    cell movement (1-19)
      if event time = clock then
        execute cell movement (m2)
      else
        reschedule the event
        execute current events (m3)
        if end of simulation, set flags and exit do
      end if
    none cell movement event (19 < cdr < 112)
      if event time = clock then
        put event into current events table
      else
        if table is empty then
          set clock = event time
          put event into current events table
      else
        reschedule the event
        execute current events (m3)
        if end of simulation, set flags, exit do
        end if
      end if
    case else
      error condition
      set error flag and exit do
    end select
end loop
```

FIG. 3. Main body of the simulation driver.

The end of simulation, or a user induced keyboard interrupt to stop execution sets up some flags that are tested at step (3). Similar to all other events except the cell movements, the end of simulation event and a stop simulation request enters into the current events table before execution. Therefore, end of simulation condition is always checked whenever the current events table is executed as shown on Fig. 3. If the event is a cell movement from one phase to another (code 1-19), module m2 is called to execute it unless there are some earlier events in the current events table.

The infinite loop is exited in several ways; by the "end of simulation" event, by a user interrupt through the keyboard or by an error condition. These conditions are recognized through flag settings that are tested right after the loop as shown on Fig. 4. If the simulation fails, a message is printed and access to temporary results files are disabled. Otherwise, the user is prompted with a question about continuing this experiment in the future. If affirmative, ending conditions of the experiments saved on the hard disk, and the experiment status and directories are updated, the files are properly closed and control is returned to the main module.

```
if simulation is successful then
    if the experiment is to be continued,
        save final conditions, update experiment
        directory and status
    update result directory
    save phase names
else
    print message
end if
close files
```

FIG. 4. Ending activities of the simulator driver.

7. Cell Movements (m2)

Cell movements are the most common events. Figure 5 shows the structure of its driver routine. Normally, a target phase is determined, the phase statistics are updated, a random phase transit time is computed and the cell is scheduled to leave the target phase at the end of the calculated time. Basically module m2 is structured in the above manner and then the details and perturbations are superimposed on it.

If there is an active block in the phase, the cell will not move to the next phase until the block is lifted. If the target phase is absorbant, or if a killing agent is present, the statistics are updated and the cell is taken out of the system. If there is labeling activity in target phase, the cell is marked. If there is a block in the target phase's entrance, the cell will be kept there until the block is lifted.

```
begin cell movement driver routine (m2)
    if the current phase is absorbant exit routine
    unpack and define event attributes
    if there is a block in current phase then
        update block statistics and mark the cell
        reschedule a cell movement at the end of block
    else
        check block status and reschedule the cell
            movements for the cells on quarter blocks
    end if
    check label and update statistics
    check proliferation and update statistics
    determine the target phase
    if the target phase is absorbant, or if the cell
        is killed or lost, update statistics and exit
    determine the phase transit time randomly
    if there is labeling in target phase, update stats
    if there is a block in target phase, mark the cell
        and schedule cell movement according to the
        block release time
    define attributes
    schedule a cell movement
end routine
```

FIG. 5. Cell movement driver routine.

8. Current Events Table Execution (m3)

All the events except the cell movements are brought into the current events table before execution. It has the same function as the current events chain in other simulation language processors. Since its size is small, a simple table is used in place of a chain. This table contains all the events that are occurring at the same time and their attributes. This mechanism is generated to resolve the execution time conflicts among the events on the basis of priority levels. Since cell movements always have the highest priority they never enter into this table. Module m3 takes care of the execution of the events in this table and uses about a dozen helpers. The structure of the driver routine is shown on Fig. 6.

If the current events table is empty, an indicator flag is set and control is returned to the calling routine which is the simulation driver. Otherwise, the events and their attributes are sorted according to priority levels and executed in that sequence as shown on Fig. 6. The first case deals with user interrupts. The execution is stopped, a message is displayed and a user reply is sought concerning what to do next.

```
begin current events driver routine.
   if table is empty, set failure flag and exit routine
   sort table on priority
   for each event in the table
      select case on event code
      21, 22 : scheduled or keyboard interrupt
               halt simulation, print message and wait for
               user input
               if "s" is entered then print message and set
                  flags for run termination
      31     : pulse kill command
      32     : activate continuous kill command
      33     : deactivate continuous kill command
      41     : freeze development command
      51     : delay development command
      61     : pulse labeling command
      62     : activate continuous labeling command
      63     : deactivate continous labeling command
      72     : activate block command
      73     : deactivate block command
      101    : screen clock update
      102    : screen graph update
      106    : collect and save information about the status
               of the experiment
      111    : end of simulation event
                  display message, set simulation
                  termination flags
      else   : unrecoverable error condition, display message
                  set error flags, dump arrays and important
                  variables onto disk for postmortem analysis
      end select
   next event
   if there has been a keyboard entry  and it is "i" then
   schedule a keyboard interrupt event immediately
   else
      ignore the keyboard entry
   end if
   clear the table
end routine
```

FIG. 6. Current events execution module driver routine.

The event codes greater than 29 and less than 100 are reserved to represent the drug effect facilities provided by the system. Event code 31 is used for instant killing of the cells in a specified phase. The system scans the future events chain, finds a cell in this phase, draws a random number between 0-1 and decides if it is killed by comparing the random number with the effectiveness rate of the kill specification. If the killing is to stay effective for a period of time, then it is handled by two events. The event code 32 initiates the instant kill effect; and then sets up some indicators to make the kill command active for the cells which will be entering the phase in the future. At the end of the effectiveness period, the event coded 33 deactivates killing by resetting the indicators. The rest of the events concerning commands work in a similar manner also. The screen clock is updated every 0.1 hour to show the progress of the simulation. The rest of the information on the screen, the number of cells in each phase and the histogram are updated every half hour after collecting information on the status of the experiment.

9. DNA Synthesis Simulation

The system imitates the new DNA formation in the synthesis phase by referring to the DNA growth curve specified during experiment description. The system provides several choices. Figure 7 shows two default options. It describes the percentage of DNA synthesis in terms of the percentage of time spent in the synthesis phase for a given cell[12, 13]. Since phase transit time is stochastic, each cell's syntheses phase duration is unique, and therefore the percentage of time spent in synthesis phase is computed with regard to this unique time. Once the percentage of time in the synthesis phase is found, the computation of the new DNA amount is straightforward. Geometrically speaking, we enter the x-axis with the given percentage, cut the curve and read the DNA amount from the y-axis. Of course, the algebraic equivalent of the above description is carried out by the system. DNA amounts in the cells are updated at information collection times, *i.e.*, every half hour. The amount of DNA in the cells preceding the synthesis phase are assumed to be 1. After the synthesis phase it is 2 until the end of the mitosis phase.

10. Experiment Status Information Collection

Every half hour, as well as at the beginning and at the end of simulation, the following information is extracted from the data structures: Clock time, report number, the number of labeled and unlabeled cells in the culture, the number of cells with DNA amount 1, with DNA amount 2, and with 1-10%, 11-20%, 21-30%, ... , 91-100% new DNA completions, the cumulative number of cells killed, frozen, delayed, blocked, lost and divided thus far. This data is written onto the first temporary results file. Phase specific information is also collected at this time for each phase in the cycle. They are as follows: The number labeled and unlabeled cells in the phase, the number of groups in the phase, the cumulative number of cells killed, frozen, delayed, labeled, blocked and lost in the phase. This data for all the phrases in the cell cycle is stored onto the second temporary results file. At the end of the simulation, the output facilities can be used to examine the data in these files, and if deemed fit, they can be saved as permanent user files.

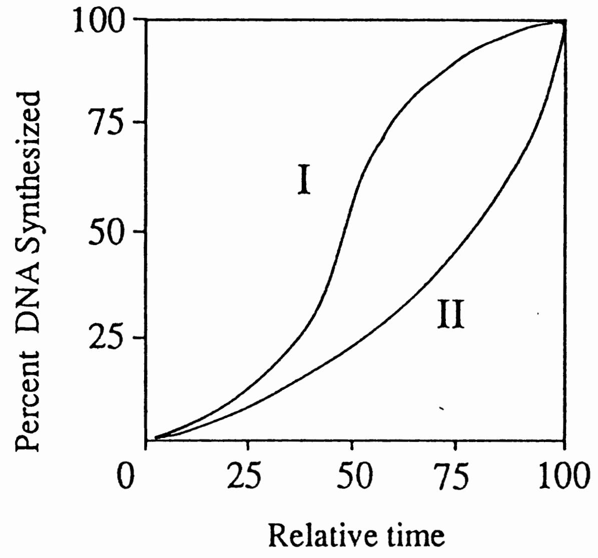

FIG. 7. Chromosomal DNA Synthesis during S-Phase.

11. Illustration

The simulator has been tested with fictitious and real data, and the results have been reported in the final report of this project[14]. We would like to present some results here in order to give a flavor about the product. The effect of the drug thymidine on the chinese hamster ovary (CHO) cells was modeled by exposing an asynchronous culture of CHO cells to thymidine at a concentration that blocked 90% of the cells and caused them to accumulate at the first quartile of S-phase. In the modeling process, the population is constructed using data available in the literature about CHO cells. The initial number of cells was set to 1000 and the cell cycle was divided into four consecutive phases, that is G1 = 2.5 hr, S = 8 hr, G2 = 1.5 hr and M = 0.5 hr. The drug was added to the asynchronous growing cells 8 hours after plating of the cells and was allowed to stay for one generation time (12.5 hr). At the end of thymidine treatment at time 20.5 hr the drug was removed and cells were allowed to continue growing in fresh media for two more generations.

Figure 8 shows the simulation of thymidine effects on the proliferation of CHO cells. After the drug removal, cells moved through the remaining period of S-phase (6 hrs) and through G2, then the actual cell divisions occurred in M and by the end of M, cells have duplicated and continued to move in the cycle in asynchronous fashion. Figures 9 and 10 show the detailed behaviour of cells in G1 and S phases. The cells continue to accumulate in G2-phase after thymidine addition. However, cells which bypass the thymidine blocking point continue to move through S, G2, M and G1 phases and then they are blocked in the first quartile of S-phase in the second cycle. After the drug removal, cells start to leave G1-phase and enter S-phase (Fig. 10) in a synchronous fashion.

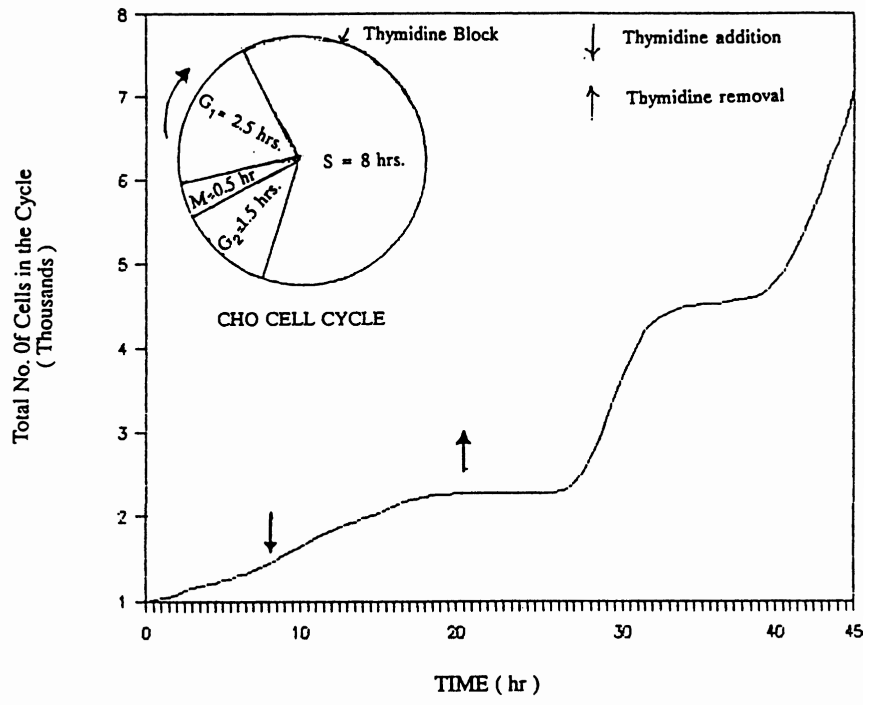

FIG. 8. Computer simulation of cellular proliferation in the presence of thymidine.

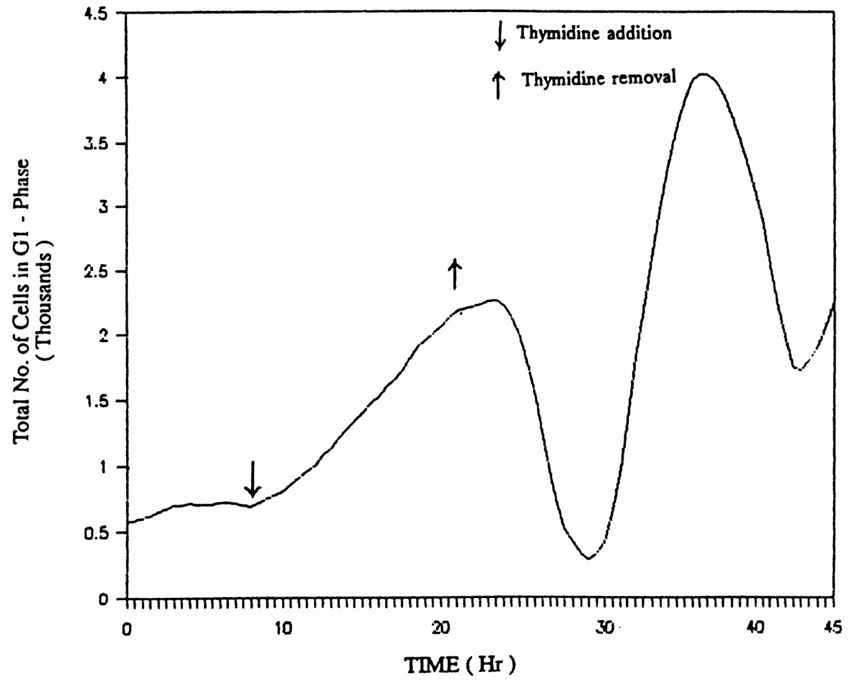

FIG. 9. Computer simulation of flow cells in G1-phase in the presence of thymidine.

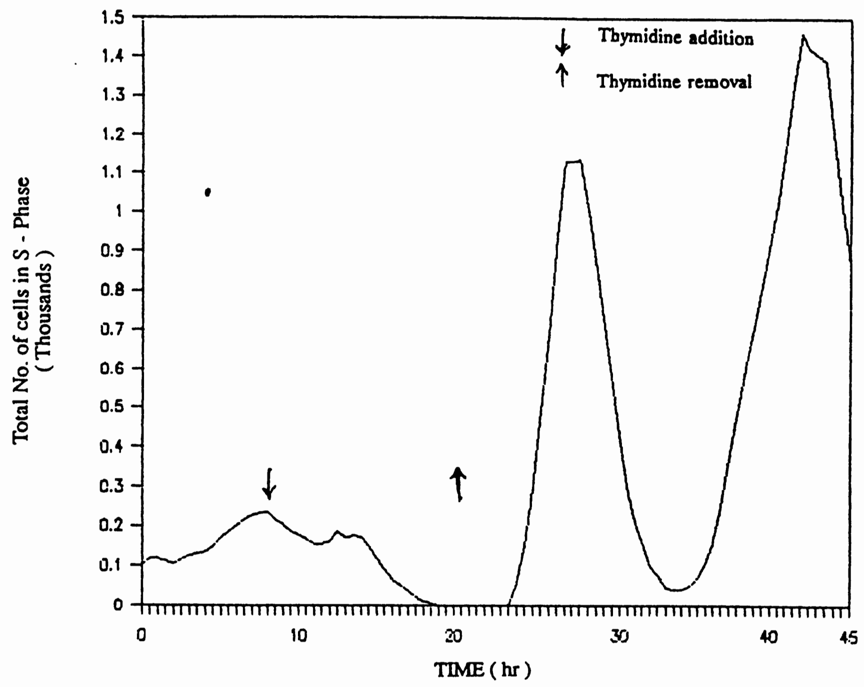

FIG. 10. Computer simulation of flow of cells in S-phase in the presence of thymidine.

12. Summary

A software package called Cell Growth Simulation (CGS) is developed to imitate growth of cell cultures under different conditions. Stochastic simulation is based on cell cycle kinetics. Experiments involving DNA synthesis inhibition, mitosis inhibition, labeling, cell kill, delaying cell maturation, blocking cell development etc., can be simulated by the system, *i.e.*, the affects of different types of agents on the cultures can be accommodated. CGS provides facilities to help the user describe, verify, save and recall the experiments easily. Experiments are simulated according to the user specifications and the results are displayed by graphs or tables. This article explains the simulator modules of CGS and how they work. This software is implemented on AT type personal computers. It contains about 6200 source lines in True Basic language.

References

[1] **Newton, C.M.**, Computer simulation of stem cell kinetics, *Bulletin of Mathematical Biophysics*, **27**: 275-290 (Special Issue) (1965).

[2] **Kember, N.F.**, Growing bones on the computer – some pitfalls of a computer simulation of the effects of radiation on growth, cell & tissue kinetics, **2**:(1) Jan.: 11-20 (1965).

[3] **Tibaux, G., Firket, H.** and **Hopper, A.F.**, A Simple Computer Simulation Model for Growing Cell Populations, *Cell & Tissue Kinetics*, **2**(4) Oct. : 333-345 (1969).

[4] **Wilson, R.** and **Gehan, E.**, A Digital Simulation of Cell Kinetics with Applications to 1210 Cells, Computer Programs in Bio-Medicine (1970).

[5] **Tsanev, R.** and **Sendlov, B.,** A model of cancer studied by a computer, *Journal of Theoretical Biology,* **23:** 124-134 (1969).

[6] **Stubblefield, E.** and **Dennis, C.M.,** The simulation of thymidine kinase kinetics in cultural cells, *Journal of Theoretical Biology,* **1:** 171-184 (1976).

[7] **Donoghey, C.E.,** CELLSIM: Cell Cycle Simulation Made Easy, *International Review of Cytology,* **66:** 171-210 (1980).

[8] **Ducting, W.** and **Vogelsaenger T.,** Three dimensional simulation of tumor growth, *Simulation,* **40**(May): 163-170 (1983).

[9] **Abed, S.Y., Ozkul, O.S., Alidrisi, M.M., El-Assouli, S.** and **Amer, M.,** Cell growth simulation (CGS): The design philosophy, *Int. J. Systems Sci.,* **24**(12): 2219-2227 (1993).

[10] **Abed, S.Y., Ozkul, O.S., Alidrisi, M.M., El-Assouli, S.** and **Amer, M.,** Cell growth simulation (CGS): A user friendly system, *Summer Computer Simulation Conference, USA, July,* pp. 658-663 (1991).

[11] **Cook, J.R.** and **Thomas, W.J.,** Age distribution of cells in logarithmically growing cell populations, *In:* **E. Zeuter (ed.),** *Synchrony in Cell Division and Growth,* Interscience Publishers, New York, pp. 485-495 (1964).

[12] **Brewer, B.J., Zakian, V.A.** and **Fangman, W.,** Replication and meicotic transmission of yeast ribosomal, *RNA genes. Proc. Natl. Acad. Sci.* **77:** 6739-6743 (1980).

[13] **Zakian, V.A., Brewer, B.J.** and **Fangman, W.L.,** Replication of each copy of the yeast 2 micron DNA Plasmid occurs during S-phase, *Cell,* **17:** 923-934 (1979).

[14] **Abed, S.Y.,** *Simulation of Normal and Cancerous Cell Populations and Drug Effects,* Final Report submitted to KACST, Riyadh, Saudi Arabia, July (1990).

نظــام محــاكاة نمــو الخــلايا : المحــاكي

عثمان شادي أوزكول[١] ، وسراج يوسف عابد[١] ، ومصطفى محمد الإدريسي[١] وسفيان العسولي[٢] ، وماجد عامر[٣]

١ *قسم الهندسة الصناعية ، كلية الهندسة ، جامعة الملك عبد العزيز ، جــدة*
٢ *قسم علوم الأحياء الطبية ، كلية الطب ، جامعة الملك عبد العزيز ، جــدة*
٣ *قسم الأورام ، مستشفى الملك فيصل التخصصي ومركز الأبحاث ، الرياض*
المملكة العربية السعودية

المستخلص . نظام محاكاة نمو الخلايا هو عبارة عن مجموعة برمجيات تحاكي نمو الخلايا في المزارع الخلوية ، ويمكن لهذا النظام محاكاة تأثير كثير من العوامل كالعقاقير على هذه الخلايا . وهو نظام محاكاة عشوائي stochastic system ، يعتمد أساسًا على حركيات الدورة الخلوية . ويمكن لهذا النظام محاكاة العديد من التجارب ، مثل تجارب تكوين الدنا (DNA) ، إيقاف أو إحصار الانقسام الخلوي الفتيلي ، وكذلك تجارب تشعيع الخلايا أو قتلها . ويتم ذلك من خلال محاكاة إضافة العقاقير ذات التأثيرات المختلفة مثل الثيميدين ، وافيديكولين ، وهيدركسي يوريا ، وفنكرستين ، وفنبلستين ، والكولسيميد إلى الخلايا . وتتم المحاكاة عن طريق وصف المستخدم للتجربة طبقًا للافتراضات النظرية ، وحسب مرئياته الموضوعية للتجربة .

ويوفر هذا النظام للمستخدم عدة تسهيلات تمكنه من وصف التجربة ، ومن ثم فحص نتائجها . ويتعامل المستخدم مع هذا النظام من خلال قوائم في شاشات مختلفة تعرض على المستخدم بطريقة أوتوماتيكية ، وجميعها موجودة على حاسب شخصي من نوع AT مزود بأسطوانة صلبة . وتركز هذه الدراسة على شرح وحدات المحاكي (simulator modules) في هذا النظام ، والتي تم تصميمها بناءً على وقوع الأحداث ، وبحيث تتبع دورة حياة كل خلية في المزرعة . كما تقوم هذه الدراسة أيضًا بشرح التركيب العام للمحاكي ، والأحداث وخصائصها ، وتصميم سلاسل الأحداث ، وتجهيز المزرعة ، بالإضافة إلى شرح عملية محاكاة نضوج (بلوغ) الخلايا ، وعملية تكوين الدنا (DNA) . هذا إلى جانب شرح عملية تجميع البيانات الإحصائية عن التجارب التي يتم محاكاتها .