სისტემების
მოდელირება და სიმულაცია

(ლექციათა კურსი - 2013)

ივ. ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტის
ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტის
კომპიუტერულ მეცნიერებათა დეპარტამენტის
გამოყენებითი ინფორმატიკის პროფესორი

გია სირბილაძე

ავტორის წინასიტყვაობა

ლექციათა კურსის "სისტემების მოდელირება და სიმულაცია"-მიზანია სტუდენტებს მისცეს ყოველმხრივი წარმოდგენა სიმულაციური მოდელირებით ბუნებაში და საზოგადოებაში მიმდინარე რთული სისტემებისა და პროცესების ევოლუციის შესწავლის ყველა მნიშვნელოვან ასპექტზე. მათ შორის კომპიუტერული მოდელირების პროგრამულ უზრუნველყოფაზე, მოდელების ადექვატურობასა და საიმედოობაზე, შემავალი მონაცემების მოდელირებაზე, შემთხვევითი რიცხვების გენერატორებზე, მოდელირებადი ექსპერიმენტების სტატისტიკურ დაგეგმარებასა და ანალიზზე, გამომავალი მონაცემების ანალიზსა და მოდელის მდგრადობაზე, სიმულაციური მოდელების ვერიფიკაციასა და ვალიდურობაზე, აგრეთვე მოდელირებისა და სიმულაციის გამოყენების ძირითადი სფეროების გაშუქებაზე, ნაწილობრივ კი საწარმოო, რიგებისა და სხვა სისტემების კომპიუტერული მოდელირების შესაძლებლობებზე.

ლექციათა კურსში შეხვდებით მრავალ მაგალითს და ამოცანას, აგრეთვე იმ ავტორთა შედეგებს (თარიღების მითითებით), რომელთა დიდი ღვაწლია კომპიუტერული სიმულაციების განვითარება.

ლექციათა კურსის სწავლების ამოცანაა:

ა) კომპიუტერულ-სიმულაციური მოდელირების ფუნქციონალური ცნების განხილვა-მსჯელობა;
ბ) კომპიუტერულ-სიმულაციური მოდელირების შეფასება;
გ) შემთხვევითი რიცხვების გენერატორების შედარება;
დ) სიმულაციური მოდელების პროექტის სახით შექმნა;
ე) კომპიუტერული პროგრამების, რომლებიც წარმოადგენენ ლექციათა კურსში ჩამოყალიბებული და წარმოდგენილი მოდელირების მეთოდებისა და ალგორითმების ილუსტრაციებს, დაპროექტების, დაპროგრამების, ტესტირებისა და გამართვის ჩატარება.

ლექციათა კურსი "სისტემების მოდელირება და სიმულაცია" შეესაბამება მსოფლიოში აღიარებული კომპიუტერული საზოგადოებების ACM-ისა და IEEE-ის Computing-ის (ინფორმატიკა ზოგადად) მეთოდოლოგიას საგანში: "მოდელირება და სიმულაცია". ლექციათა კურსი ეფუძნება საგნის მსოფლიოში აღიარებულ სახელმძღვანელოებს [1-6].

ლექციათა კურსი განკუთვნილია Computing-ის (ინფორმატიკის) ყველა სპეციალობებისთვის, მათ შორის "კომპიუტერული მეცნიერება"; აგრეთვე მათემატიკის, განსაკუთრებით გამოყენებითი მათემატიკის, ფიზიკის, ბიოლოგიის, სოციოლოგიის, ბიზნესის, წარმოების ტექნოლოგიების სწავლების სტუდენტებისთვის, თუ ისინი შეასრულებენ საგნის შესწავლის წინაპირობებს:

1. დისკრეტული ალბათობისა და სტატისტიკის საფუძვლები;
2. გამოთვლითი ტექნიკის გარკვეული ცოდნა (ინფორმატიკის საფუძვლები);

3. უნივერსალური დაპროგრამების ენა C-ს ცოდნა;
4. წრფივი ალგებრისა და მატრიცული აღრიცხვების ზოგიერთი საკითხები.

ლექციათა კურსის "სისტემების მოდელირება და სიმულაცია" გავლის შემდეგ სტუდენტს შეეძლება სისტემების, მათ შორის რთული სისტემების სიმულაციური მოდელირებით სრული და ეფექტური შესწავლა-ანალიზი. ასევე მზად იქნება შეისწავლოს აღნიშნული კურსის უფრო რთული კურსი მაგისტრატურასა და შესაძლოა დოქტორანტურაშიც.

ლექციათა კურსი "სისტემების მოდელირება და სიმულაცია" კურსსთან "ოპერაციათა კვლევა" შეიძლება ჩაითვალოს Computing-ის სპეციალობების ფუნდამენტურ სამუშაო და კვლევით ინსტრუმენტად.

ლექციათა კურსის "სისტემების მოდელირება და სიმულაცია" შესწავლისას ძირითადი და დამხმარე მასალების სარგებლობისათვის (დიდი რაოდენობა ამოცანების ამოხსნები, ლექციათა სლაიდები და ამოცანებისა და ალგორითმების პროგრამული კოდები) სტუდენტებმა შეიძლება ჩაიხედონ ელ. გვერდზე www.mhhe.com/lawkelton. მასალასთან წვდომის რეკვიზიტების მისაღებად დაუკავშირდით გამომცემლობა McGraw-Hill-ის წარმომადგენელს ელ. მისამართზე: mhcom@mcgraw-hill.com ან ზემოთ მოყვანილი ელ. გვერდის საშუალებით.

"O dear Ophelia"
"I am ill at these numbers:
"I have not art to reckon my goans."
შექსპირი: "ჰამლეტი" (II მოქმედება, II სცენა)
(ო, ძვირფასო ოფელია! მე ავად ვარ ამ რიცხვებით:
მე არ შემიძლია დავითვალო ჩემი სუნთქვა)

“მრგვალი რიცხვები ყოველთვის ყალბია”

სემიუელ ჯონსონი (1750)

თავი 1. მოდელირებისა და სიმულაციის შესახებ

შესავალი

სასწავლო კურსში განიხილება კომპიუტერული ტექნოლოგიების გამოყენება სხვადასხვა სისტემებისა თუ პროცესების ან მოწყობილობების ოპერაციების სიმულიციისთვის (ანუ ხელოვნური მოდელირებისთვის). მოწყობილობა თუ პროცესი ჩვეულებრივ იწოდება *სისტემად*. სისტემების შესწავლისა თუ ფუნქციონირების (ევოლუციის) დასადგენად ჩვენ ვაკეთებთ გარკვეულ დაშვებებს, რომელიც ეხება სისტემის ფუნქციონირებას.

როგორც ყოველთვის ეს დაშვებები ჩაიწერება *მათემატიკურ-ლოგიკურ მიმართებებში*, რომლებიც ყალიბდება გარკვეულ სტრუქტურაში – *მოდელში*, რომლის დახმარებით შეიძლება მივიღოთ წარმოდგენა შესაბამისი სისტემის ქცევაზე.

თუ მიმართებები, რომლებშიც ყალიბდება სისტემის მოდელი, საკმაოდ მარტივია ჩვენთვის საჭირო ინფორმაციის მისაღებად სისტემის ფუნქციონირებაზე, მაშინ მიზნის მისაღწევად საკმარისია მათემატიკურ-ანალიზური მეთოდებით სარგებლობა. ასეთ მოდელებს *ანალიზური მოდელები* ეწოდებათ. თუმცა უმრავლესობა სისტემებისა რთულია, და მათთვის შეუძლებელია ისეთი რეალური მოდელის შექმნა, რომლის რეალიზაცია, ამოხსნა ჩაიწერება, წარმოიდგინება ანალიზურად. ასეთი სისტემები და მათთვის შექმნილი მოდელები როგორც ყოველთვის შეისწავლება *მოდელირებით*. სისტემის მოდელირებისას აუცილებლად გამოიყენება *კომპიუტერი*, მისით *მოდელის რიცხვითი შეფასებისთვის*. ხოლო მიღებული მონაცემებით კი უკვე გაითვლება *სისტემის რეალური მახასიათებლები*, რომლის მიხედვითაც უკვე *წინასწარ კეთდება დასკვნები* სისტემის ქცევის, მისი ევოლუციის ან ფუნქციონირების შესახებ.

მოვიყვანოთ მაგალითი:

მოდელირება შეიძლება გამოყენებული იყოს მაგალითად საწარმოო კომპანიისთვის, რათა შეფასდეს შესაძლებლობა მისი ერთ-ერთი ქარხნისთვის დამატებითი დიდი სიმძლავრეებისა და შენობების ასაშენებლად. თუმცა ხელმძღვანელობა-ადმინისტრაციული კორპუსი არაა დარწმუნებული, რომ წარმოების პოტენციურმა ზრდამ შეიძლება გაამართლოს მშენებლობებზე გაწეულ დანახარჯებთან მიმართებაში. შეუძლებელია ჯერ ააშენო, შექმნა სიმძლავრეები, შენობა-ნაგებობანი, ხოლო შემდეგ კი არა რენტაბელობის გამო ავიღოთ და “დავანგრიოთ” ისინი. ამ დროს კი მიმდინარე მომენტში ქარხნის მუშაობის მოდელირება “თითქოს და” შექმნილი დამატებითი სიმძლავრეებით (ხელოვნურად, სიმულაციით წარმოქმნილი სიმძლავრეებით) ეხმარება ადმინისტრაციას ამ პრობლემის გადაწყვეტაში.

მოდელირება და სიმულაცია გამოყენებული შეიძლება იყოს ადამიანის საქმიანობის სხვადასხვა სფეროებში. იგი განსაკუთრებით ეფექტურია შემდეგი პრობლემების გადაწყვეტაში:

ა) წარმოების და ბიზნეს სისტემების პროექტირება და ანალიზი;

ბ) სამხედრო შეიარაღების სხვადასხვა სისტემების შეფასება და მოთხოვნების დაგეგმვა მათი მატერიალურ-ტექნიკური უზრუნველყოფისათვის;

გ) კავშირის ქსელების პროტოკოლებისა და მოწყობილობებისთვის მოთხოვნის განსაზღვრა;

დ) სხვადასხვა კომპიუტერული სისტემების პროგრამული უზრუნველყოფისა და მოწყობილობებზე მოთხოვნის განსაზღვრა;

ე) სატრანსპორტო სისტემების პროექტირება და მუშაობის ანალიზი. მაგალითად: აეროპორტები, ავტომაგისტრალები, ქალაქის საკომუნიკაციო-საავტომობილო სისტემები, მეტროპოლიტენი და სხვ.

ვ) სხვადასხვა მასობრივი მომსახურეობის ორგანიზაციების შექმნის პროექტების შეფასება. მაგალითად: შეკვეთების დამუშავების ცენტრი, სწრაფი კვების სისტემა, საავადმყოფოების, ჰოსპიტალების ქსელი, სხვადასხვა მომსახურეობის, თუ კავშირის ქსელები, პროვაიდერები და სხვ.

ზ) საქმიან სფეროში სხვადასხვა პროცესების მოდელირება;

თ) რესურსების მართვის სისტემებში პოლიტიკის განსაზღვრა;

ი) ფინანსური და ეკონომიკური სისტემების წინასწარი ანალიზი;

კ) ეკო-სისტემები და გლობალური დათბობის პროცესების სიმულაციური მოდელირება და სხვ.

ბოლოს შევნიშნოთ, რომ რაც დრო გადის სისტემები უფრო და უფრო რთულდება და ანალიზური მოდელები პრაქტიკულად გამოუყენებელი ხდება. ბოლო პერიოდში სიმულაციური მოდელირების გამოყენება და გავრცელება უკავშირდება რამდენიმე ფაქტორი-პრობლემის ფაქტიურ დაძლევას:

ა) მოდელებისთვის, რომლებიც ბოლო პერიოდში რთულ სისტემებსა და პროცესებს იკვლევენ, გართულებულია კომპიუტერული პროგრამების დაწერა, თუმცა უკანასკნელ წლებში ეს პრობლემა იხსნება ძლიერი პროგრამული პროდუქტების გამოჩენასთან ერთად;

ბ) რთული სისტემების მოდელირება ხშირად ითხოვს დიდ მანქანურ დროს. თუმცა უკანასკნელი ძლიერი კომპიუტერების სწრაფქმედების ზრდითა და თვითღირებულების შემცირებით ეს პრობლემაც თანდათანობით იხსნება;

გ) კომპიუტერულ მოდელირებაზე ხშირად იქმნება არასწორი შთაბეჭდილება, როგორც დაპროგრამების სავარჯიშოზე [1], თუნდაც როგორი რთულიც არ უნდა იყოს იგი. ამ შეხედულების საფუძველზე კვლევები სისტემებზე მოდელირებით სრულდება, როგორც ევრისტიკული მოდელის დაპროგრამება, მისი შემდგომი ერთჯერადი ‘‘გაშვებით’’ პროგრამაზე ‘‘პასუხების’’ მისაღებად. ასეთი მიდგომა აზრს კარგავს სისტემების სირთულის ზრდასთან ერთად, რასაც თვითონ დაპროგრამების მსოფლიოს წამყვანი ჯგუფებიც ადასტურებენ [1-3].

§1.1. სისტემა, მოდელი და სიმულაციური მოდელირება

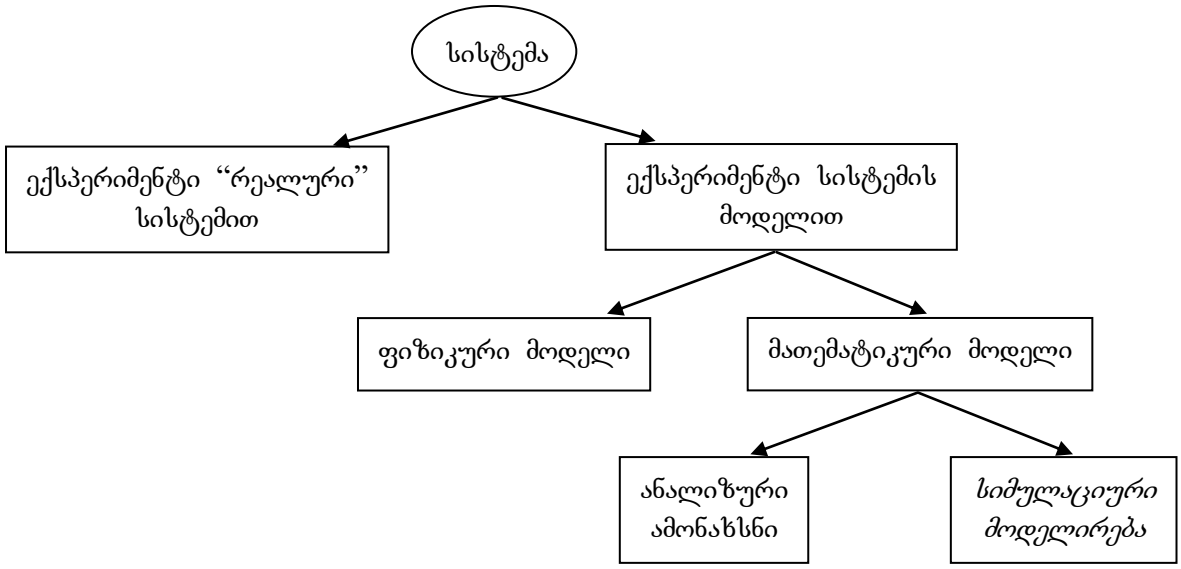
სისტემა წარმოადგენს ობიექტების, მაგალითად ადამიანების ან მექანიზმების ერთობლიობას, რომლებიც ფუნქციონირებენ და ერთმანეთთან ურთიერთქმედებენ გარკვეული მიზნის მისაღწევად (Schmidt and Taylor, 1976). პრაქტიკაში სისტემის ცნება დამოკიდებულია კონკრეტული კვლევების ამოცანებზე. მაგალითად, ობიექტების ერთობლიობა, რომლებიც ქმნიან სისტემას ერთი კვლევისთვის, შეიძლება წარმოადგენდნენ მხოლოდ ქვესიმრავლეს მეორე სისტემაში, სადაც კვლევის ამოცანა უკვე სხვაა და შეიძლება უფრო გლობალურიც.

სისტემის მდგომარეობა განისაზღვრება, როგორც დროის განსაზღვრულ მომენტში სისტემის აღწერისათვის აუცილებელი ცვლადების მნიშვნელობათა ერთობლიობა, რომელიც შეესაბამება კვლევის ამოცანებს. მაგალითად ბანკის კვლევისთვის მდგომარეობის ცვლადებად შეიძლება ჩაითვალოს: დაკავებული მოლარეების რაოდენობა, ბანკში კლიენტთა რაოდენობა და მათი ბანკში მისვლის დრო და სხვა.

არსებობს ორი ტიპის სისტემა: *დისკრეტული* და *უწყვეტი*. დისკრეტული სისტემაში დროის განსხვავებულ პერიოდში სისტემის მდგომარეობის ცვლადები იცვლებიან მყისიერად. ბანკი შეიძლება ჩაითვალოს დისკრეტულ სისტემად, მისი პარამეტრების ბუნების გამო. *უწყვეტ სისტემებში* ცვლადები დროში იცვლებიან უწყვეტად. მაგალითად, თვითმფრინავი, როგორც სისტემა, რომელიც მოძრაობს ჰაერში, უწყვეტი სისტემის მაგალითია, რადგან მდგომარეობის ცვლადები (თვითმფრინავის მდებარეობა-კოორდინატები, სიჩქარე და სხვ.) იცვლებიან უწყვეტად დროის ცვლილებასთან ერთად. პრაქტიკაში სისტემა იშვიათადაა

სრულიად დისკრეტული ან უწყვეტი, თუმცა ძირითადად სისტემაში დომინირებენ ერთი ან მეორე ტიპის ცვლადები და იმის მიხედვით სისტემას მიაკუთვნებენ დისკრეტულს ან უწყვეტს. ჩვენი კურსის განხილვის საგანია დისკრეტული სისტემები და მათი მოდელირება.

უმრავლესობა სისტემებისთვის ფუნქციონირების განსაზღვრულ მომენტებში, აუცილებელია კვლევის ჩატარება მისი კომპონენტების ურთიერთქმედებისა და ექსპლუატაციის ახალ პირობებში წარმოდგენების მიღების მიზნით. ნახაზ 1.1-ზე გამოსახულია სისტემის კვლევის სხვადასხვა საშუალებები:



ნახ. 1.1

განვიხილოთ ნახაზის თითოეული ობიექტი:

1. *ექსპერიმენტი რეალური სისტემით თუ ექსპერიმენტი სისტემის მოდელით?* როდესაც არსებობს შესაძლებლობა ფიზიკურად შეიცვალოს სისტემა (თუ იგი რენტაბელურია) და იგი ახალ პირობებში მოვიყვანოთ მოქმედებაში. ეს საუკეთესო შესაძლებლობაა და ასეც უნდა მოვიქცეთ, რადგან საკითხი მიღებული რეზულტატების ადექვატურობაზე თავის თავად მოხსნილია. თუმცა ასეთი მიდგომა ხშირად განუხორციელებელია მისი შესრულების დიდი დანახარჯების გამო ან თვითონ სისტემაზე დამანგრეველი ზემოქმედების გამო. მაგალითად: ბანკში ეძებენ გზებს დანახარჯების შესამცირებლად. ამ მიზნით სთავაზობენ მოლარეების რიცხვის შემცირებას. თუ გამოვცდით ახალ სისიტემას, რომელშიც მოლარეთა რიცხვი შემცირებულია, შეიძლება მივიღოთ კლიენტთა გრძელი რიგები და დაგვიანება მომსახურეობაზე და ამით კი ბანკის მიერ კლიენტების დაკარგვა; უფრო მეტიც, სისტემა შეიძლება სინამდვილეში არც არსებობდეს, მაგრამ საჭირო იყოს მისი სხვადასხვა კონფიგურაციების შესწავლა, იმისთვის რომ ავურჩიოთ შესრულების ყველაზე ეფექტური საშუალება. ასეთი სისტემების მაგალითებია კავშირის ქსელები ან ბირთვული შეიარაღების სტრატეგიული სისტემები. ამიტომ აუცილებელია *სისტემის მოდელების შექმნა*, რათა გამოვიკვლიოთ იგი, როგორც *რეალური სისტემის შემცვლელი*. რეალური სისტემის სანაცვლოდ მისი მოდელით სარგებლობისას ყოველთვის წარმოიქმნება შეკითვა: ნამდვილად კი მოდელი ისეთი ხარისხით ასახავს რეალურ სისტემას, რომ მივიღოთ გადაწყვეტილება, რომელიც დაეფუძნება მოდელურ კვლევებს (მოდელის ვარიაბელობისა და ვალიდურობის საკითხს მოგვიანებით განვიხილავთ).

2. *ფიზიკური მოდელი თუ მათემატიკური მოდელი*? სიტყვა "მოდელი" უმრავლესობას ჩვენს შორის გონებაში წარმოუდგენს, მაგალითად, კაბინებს, რომლებიც დამონტაჟებულია სავარჯიშო მოედნებზე, ვთქვათ მფრინავების სწავლებისთვის; მინიატურულ ტანკერებს, რომლებიც მოძრაობენ ბასეინში და სხვა. ყოველივე ეს *ფიზიკური მოდელების* მაგალითებია. ისინი იშვიათად გამოიყენება სისტემების ანალიზისა და ოპერაციების კვლევაში. თუმცა

ზოგიერთი ტექნიკური ან მართვის სისტემების კვლევისას საკმაოდ ეფექტურია სწორედ ფიზიკური მოდელების შექმნა. თუმცა უმრავლესობა კონსტრუირებული მოდელებისა წარმოადგენს *მათემატიკურ მოდელს.* ისინი წარმოდგებიან, როგორც სისტემა ლოგიკური და რაოდენობრივი მიმართებებისა, რომლებიც შემდგომში ექვემდებარება დამუშავებასა და ცვლილებებს, რათა განისაზღვროს თუ სისტემა როგორ რეაგირებს ცვლილებებზე, უფრო ზუსტად კი როგორი იქნება სისტემის რეაგირება, თუ ის მართლა იარსებებდა რეალობაში! ალბათ მათემატიკური მოდელის ყველაზე ელემენტარულ მაგალითს წარმოადგენს წრფივი თანაბარი მოძრაობის დამოკიდებულება: $s = v \cdot t$, სადაც s -მანძილია, v -სიჩქარე, ხოლო t -გადაადგილების დროა: ხანდახან ასეთი მოდელი შეიძლება ადექვატურიც იყოს (სხვა პლანეტისკენ მიმართული კოსმოსური ზონდის შემთხვევა), მაგრამ სხვა სიტუაციებში იგი რეალობას არ შეესაბამება (ქალაქის გადატვირთულ ქუჩებში (პიკის საათში) სატრანსპორტო გადაადგილება).

3. *ანალიზური ამონახსნი თუ სიმულაციური მოდელირება?* როცა საუბარია სისტემაზე, რომელსაც წარმოადგენს მათემატიკური მოდელი უნდა დადგინდეს თუ როგორ აიგება ეს მოდელი. როდესაც მოდელი საკმაოდ მარტივია, მაშინ შესაძლებელია მივიღოთ ან გამოვთვალოთ მიმართებები და პარამეტრები და აქედან კი მივიღოთ ზუსტი ანალიზური ამონახსნი! თუ მაგალითში, ფორმულით $s = v \cdot t$, ცნობილია მანძილი და სიჩქარე, მაშინ დრო, რომელიც საჭიროა ამ მანძილის დასაფარად გამოითვლება თანაფარდობით $t = s / v$. ეს მარტივი ზუსტი ანალიზური ამონახსნია. ზოგჯერ ანალიზური ამონახსნი საკმაოდ რთულია და ითხოვს კომპიუტერის დიდ რესურსებს. მრავალ მაგალითში ზუსტი ამონახსნი ჩაიწერება გამოსახულებით, რომელიც შეიცავს დიდი განზომილების მქონე მატრიცის შებრუნებულს. მაგრამ რომ მივიღოთ შებრუნების კარგი რიცხვითი შეფასება ამოცანა არც ისე მარტივია. თუ მათემატიკური მოდელით შესაძლებელია სისტემის ზუსტი ანალიზური ამონახსნის მიღება და მისი გამოთვლები ეფექტურია, მაშინ უკეთესია სისტემა გამოვიკვლიოთ ანალიზური გზით, ვიდრე გამოვიყენოთ სიმულაცია და მოდელირება. თუმცა მრავალი სისტემები ძალიან რთულია. ისინი პრაქტიკულად *სრულიად გამორიცხავენ* ანალიზური ამონახსნის არსებობის შესაძლებლობას. ასეთ შემთხვევაში სისტემის მოდელის შესწავლა უნდა განხორციელდეს *სიმულაციური მოდელირებით.* რაც გულისხმობს მოდელის (მათემატიკური მოდელის) *მრავალჯერად ექსპემენტის ჩატარებას* საჭირო შემავალი მონაცემებით, რათა განისაზღვროს მათი გავლენა სისტემის მუშაობის შეფასების გამომავალ კრიტერიუმებზე.

სიმულაციური მოდელირება მკვლევარის მიერ აღიქმება, როგორც "უკანასკნელი იმედის მეთოდი" (ან ხუმრობით (ავტ.) "უკანასკნელი ცრემლის სადგური") და ამაში არის ჭეშმარიტება. თუმცა რეალობაში ძალიან მალე მივდივართ იმ აზრზე რომ სიმულაცია და მოდელირება ხშირად ერთადერთი ინსტრუმენტია რთული სისტემის შესწავლისა და კვლევისთვის.

დავუშვათ, რომ ჩვენ გვაქვს მათემატიკური მოდელი, რომელიც შესწავლილი უნდა იქნას მოდელირებით (მომავალში *სიმულაციური მოდელირება* ხშირად შეიცვლება სიტყვა-*მოდელირებით*). უპირველეს ყოვლისა ჩვენთვის აუცილებელია მისი კვლევის საშუალების განსაზღვრა. ამასთან დაკავშირებით საჭიროა სიმულაციური *მოდელების კლასიფიკაცია* სამი ასპექტის მიხედვით:

1. *სტატიკური თუ დინამიკური? სტატიკური სიმულაციური* მოდელი – ეს სისტემაა დროის განსაზღვრულ მომენტში, ან კიდევ ისეთი სისტემა, რომელშიც დრო არ თამაშობს არავითარ როლს. სტატიკური სიმულაციური მოდელის მაგალითია მოდელი, რომელიც იქმნება *მონტე-კარლოს მეთოდით* [6] განსაზღვრული ჯერადი ინტეგრალების მიახლოებითი გამოთვლისთვის და ა.შ. *დინამიკური სიმულაციური მოდელი* წარმოადგენს სისტემას, რომელიც იცვლება დროში, როდესაც სისტემის მდგომარეობა ევოლუციას განიცდის დროში. მაგალითად კონვეირული სისტემა ქარხანაში. გახსოვდეთ! რომ ავაგოთ რთული სისტემის მათემატიკური მოდელი, უნდა გადავწყვიტოთ, თუ როგორ უნდა გამოვიყენოთ იგი, რომ მივიღოთ მონაცემები სისტემაზე, რომელიც შეფასებისა და კვლევისათვის აუცილებელია.

2. *დეტერმინისტული თუ სტოქასტური (ალბათურ-სტატისტიკური)?* თუ სიმულაციური მოდელი არ შეიცავს ალბათურ-სტატისტიკურ (შემთხვევით) კომპონენტებს, მას *დეტერმინისტული* ეწოდება. ასეთი მოდელის მაგალითია რთული (და ანალიზურად რთულად გამოთვლადი) დიფერენციალურ-სხვაობიანი განტოლებათა სისტემა, რომელიც აღწერს მაგალითად ქიმიურ რეაქციას-სისტემას. დეტერმინისტულ მოდელში *შედეგი შეიძლება მივიღოთ*, როდესაც მისთვის მოცემულია ყველა შემავალი ცვლადები და დამოკიდებულებანი, მაშინაც კი, როდესაც მოითხოვება კომპიუტერული დროის დიდი რაოდენობა. მაგრამ ძალიან ბევრი სისტემა მოდელირდება რამოდენიმე შემთხვევითი შემავალი კომპონენტების მონაცემებით. შედეგად იქმნება *სტოქასტური სიმულაციური მოდელი.* მასობრივი მომსახურეობისა და რესურსების მართვის უმრავლესობა სისტემებისა წარმოდგება სტოქასტური სიმულაციური მოდელებით (სსმ). სსმ შედეგად გვაძლევს შემთხვევითი სიდიდეების რეალიზაციებს, რომლებითაც სტატისტიკური შეფასების მეთოდებით შეიძლება მივიღოთ მოდელის რეალური მახასიათებლების შეფასებები.

3. *უწყვეტია თუ დისკრეტულია?* ზოგადად შეიძლება ითქვას, რომ დისკრეტული და უწყვეტი მოდელები ისევე განიმარტება, როგორც ადრე წარმოდგენილი დისკრეტული და უწყვეტი სისტემები. აქ შევნიშნოთ, რომ დისკრეტული მოდელი ყოველთვის არ გამოიყენება დისტრეტული სისტემების მოდელირებისათვის და პირიქით. ყველაფერი დაკავშირებულია *სისტემის კვლევის ამოცანაზე.* მაგალითად: ავტომაგისტრალზე სატრანსპორტო ნაკადის მოდელი იქნება დისკრეტული თუ მოდელში აუცილებელია გავითვალისწინოთ ყოველი მოძრავი მანქანის მახასიათებლები და გადაადგილება. ხოლო თუ მანქანებს განვიხილავთ ერთობლიობაში, მაშინ სატრანსპორტო ნაკადი შეიძლება აღიწეროს უწყვეტი მოდელის დიფერენციალური განტოლებით.

სიმულაციური მოდელები, რომელთაც ჩვენ შემდგომში განვიხილავთ, იქნება *დისკრეტული-ხდომილებითი დინამიკური და სტოქასტური*, რაც განპირობებული Computing-ის სპეციალობების სწავლებაში საგანის "მოდელირება და სიმულაცია" აპრობირებული მეთოდოლოგიით (ACM და IEEE-ს მეთოდოლოგია).

§1.2. დისკრეტულ-ხდომილებითი მოდელირება

დისკრეტულ-ხდომილებითი მოდელირება გამოიყენება ისეთი მოდელების ასაგებად, რომელიც ასახავს სისტემის დროში ევოლუციას, განვითარებას, როდესაც ცვლადების მდგომარეობა იცვლება *მყისიერად დროის კონკრეტულ მომენტებში* (სისტემა მდგომარეობას იცვლის მხოლოდ სასრულ ან თვლად რაოდენობა დროის მომენტებში). ასეთ მომენტებში ხდება (დადგება) *ხდომილება მყისიერად*, რომელსაც შეუძლია შეცვალოს სისტემის მდგომარეობა. თეორიულად შესაძლებელია დისკრეტული-ხდომილებითი მოდელირების შესრულება "ხელის" გამოთვლებით, თუ მიღებულ მონაცემებს წარმოვადგენთ მოხერხებული ჩანაწერებით (იხ. სავარჯიშოები), თუმცა რეალური რთული სისტემების მოდელირება ითხოვს კომპიუტერის გამოყენებას.

მაგალითი 1.1. განვიხილოთ სისტემა მომსახურეობის ერთი კვანძით (მოწყობილობით, არხით), რომელიც აღწერს სისტემის განვითარებას დროში. მაგალითად საპარიკმახერო ან ცნობათა ბიურო აეროპორტში მხოლოდ *ერთი* ოპერატორით და ა.შ. ჩვენი ამოცანაა მიახლოებით გამოვთვალოთ *რიგში მოთხოვნის დაკავების მოსალოდნელი მნიშვნელობა.* ამასთან რიგში მოთხოვნის დაკავება ტოლია იმ დროის, რომელმაც გაიარა სისტემაში მისი მოთხოვნის გამოჩენიდან მომსახურეობის დაწყებამდე. იმისთვის, რომ დავთვალოთ რიგში მოთხოვნის საშუალო დაკავება დისკრეტულ-ხდომილებით სიმულაციურ მოდელებში, აუცილებელია მდგომარეობის ისეთი ცვლადების განსაზღვრა, როგორიცაა მომსახურეობის კვანძის მდგომარეობა (მომსახურეობით დაკავებულია თუ თავისუფალია), რიგში მოთხოვნათა რიცხვი (თუ რა თქმა უნდა რიგი არსებობს) და ყოველი მოთხოვნის დადგომის დრო. მოთხოვნის დადგომისას აუცილებლად უნდა განისაზღვროს კვანძის მდგომარეობა: კვანძი მოთხოვნას მოემსახურება მოთხოვნისთანავე თუ

მოთხოვნა ჩავაყენოთ რიგის ბოლოში? მოთხოვნაზე მომსახურეობის დასრულებისას, გამომდინარე იქიდან თუ რამდენი მოთხოვნაა რიგში, უნდა განისაზღვროს კვანძი თავისუფალია მომსახურეობაზე თუ ის მოემსახურება რიგში პირველ მოთხოვნას? რომ გამოვთვალოთ რიგში მოთხოვნის დაკავება, აუცილებელია დავადგინოთ მისი მოთხოვნის მომენტი, რადგან დაკავება ტოლია მოთხოვნის მომსახურეობის დროის მომენტისა და მისი დადგომის მომენტის სხვაობის. სისტემაში არის ორი ტიპის ხდომილება: მოთხოვნის დადგომა და მოთხოვნაზე მომსახურეობის დასრულება. მოთხოვნის დადგომა ხდომილებაა, რადგან ის იწვევს მომსახურეობის კვანძის მდგომარეობის შეცლას: თავისუფალი მდგომარეობიდან დაკავებულ მდგომარეობაზე გადასვლას ან რიგში მოთხოვნების რიცხვის 1-ით გაზრდას. ამავე დროს მოთხოვნის წასვლა (გასვლა სისტემიდან) ასევე ხდომილებაა, რადგან ის იწვევს ცვლილებას კვანძის დაკავებული მდგომრეობიდან თავისუფალ მდგომარეობაზე გადასვლას ან რიგში მოთხოვნების რიცხვის 1-ით შემცირებას. მე-4-ე თავში ჩვენ ავაგებთ ამ მაგალითში წარმოდგენილი-მომსახურეობის სისტემის დისკრეტულ-ხდომილებითი სიმულაციურ მოდელს.

§1.2.1. დროის მიმდინარეობის მექანიზმი სიმულაციურ მოდელირებაში

დისკრეტულ-ხდომილებითი სიმულაციური მოდელების დინამიკური ბუნება ითხოვს, რომ თვალყური ვადევნოთ *სიმულაციური დროის მიმდინარე მნიშვნელობებს* მოდელის ფუნქციონირების გასწვრივ. ასევე აუცილებელია სიმულაციური დროის (ერთი მნიშვნელობიდან მეორე მნიშვნელობაში გადასვლის) მიმდინარეობის მექანიზმის არსებობა. სიმულაციურ მოდელში ცვლადი, რომელიც უზრუნველყოფს მოდელური დროის მიმდინარე მნიშვნელობას, *მოდელური დროის საათი* ეწოდება. მოდელების დაპროგრამების ისეთ უნივერსალურ ენებზე შედგენისას, როგორიცაა C, მოდელური დროის საათის დროის ერთეული, ცხადად არასდროს არ მიეთითება. იგულისხმება, რომ ის მიუთითებს იმავე ერთეულებში, როგორც შემავალი პარამეტრები. ამავე დროს მოდელური დრო და დრო, რომელიც აუცილებელია კომპიუტერზე სიმულაციური მოდელის გასატარებლად, როგორც წესი შეუძლებელია რაიმე თანაფარდობაში იყოს.

არსებობს მოდელური დროის მიმდინარეობის ორი მიდგომა: ა) *ხდომილებიდან ხდომილებამდე დროის მიმდინარეობა* და ბ) *დროის მიმდინარეობა მუდმივი ბიჯით*. რადგან პირველი მიდგომა გამოიყენება თითქმის ყველა ძირითად დაპროგრამების პროდუქტებში, რომლებიც გამოიყენება სიმულაციურ მოდელირებაში, ჩვენ განვიხილავთ მხოლოდ ა) ტიპის მიმდინარეობას. ამ შემთხვევაში მოდელირების საწყის მომენტში მოდელური დროის საათი ყენდება 0-ვან მნიშვნელობაზე და ხდება შემდეგი ხდომილების მოხდენის (წარმოქმნის) დროის განსაზღვრა. ამის შემდგომ მოდელური დროის საათი (მდს) *გადადის* უახლოესი ხდომილების წარმოქმნის დროზე.

მდს-ის ამ ჩვენებაზე ხდება სისტემის მდგომარეობის განახლება. ამის შემდეგ მდს-ის დროის ჩვენება გადადის შემდეგი (ახალი) უახლოესი ხდომილების წარმოქმნის დროზე. სისტემის მდგომარეობა შესაბამისად განახლდება და ა.შ. მდს-ს დროის მიმდინარეობა წყდება იმ მომენტისთვის, როდესაც შესრულდება *რაიმე წინა პირობა* სისტემის ევოლუციაზე.

რადგან აქ წარმოდგენილ მოდელში სისტემის მდგომაროებათა ცვლილებები ხდება *მხოლოდ ხდომილებათა წარმოქმნის დროს, ამიტომ სისტემის უმოქმედობის პერიოდები* უბრალოდ გამოიტოვება და საათი გადაიყვანება ერთი ხდომილების წარმოქმნის დროიდან მეორე ხდომილების წარმოქმნის დროზე.

მაგალითი 1.1 (გაგრძელება). ამ მაგალითში დეტალურად განიხილება ხდომილებიდან ხდომილებამდე დროის მიმდინარეობის მექანიზმი მასობრივი მომსახურეობის სისტემაში ერთი კვანძით (მოწყობილობით), რომელიც წინა მაგალითში უკვე განიხილებოდა. ჩვენ დაგვჭირდება შემდეგი აღნიშვნები:

t_i - i -ური მოთხოვნის დადგომის დრო $(t_0 = 0)$;

$A_i = t_i - t_{i-1}$ - $(i-1)$ და i -ური მოთხოვნების დადგომებს შორის დრო;

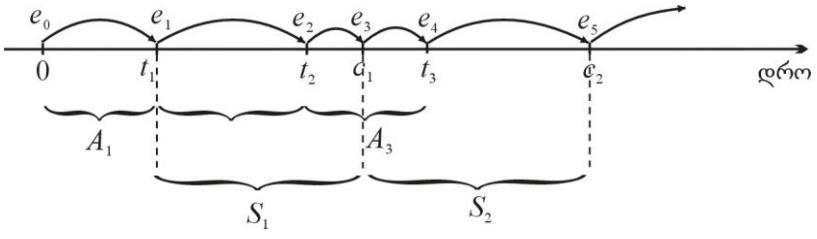
s_i -დრო, რომელიც კვანძის მიერ იხარჯება i -ური მოთხოვნის მომსახურეობაზე (რიგში მოთხოვნის დაკავების დროის გათვალისწინების გარეშე);

D_i -რიგში i -ური მოთხოვნის დაკავების დრო;

$c_i = t_i + D_i + S_i$ - i -ური მოთხოვნის მომსახურეობაზე დასრულების დრო;

e_i -ნებისმიერი ტიპის i -ური ხდომილების წარმოქმნის დრო (რომელსაც ღებულობს მდს, $e_0 = 0$ მნიშვნელობის გარეშე).

ყოველი განსაზღვრული პარამეტრი როგორც ყოველთვის შემთხვევითი სიდიდეა. დავუშვათ, რომ ჩვენთვის ცნობილია მომდევნო მოთხოვნათა დადგომებს შორის $A_1, A_2, \ldots,$ დროების ალბათური განაწილება; ასევე მომსახურეობის დროების $S_1, S_2, \ldots, \ldots$ განაწილება. მათი განაწილებათა ფუნქციები აღვნიშნოთ შესაბამისად F_A და F_S -ით. F_A და F_S განაწილებათა დასადგენად გამოიყენება ჩვენთვის ცნობილი სტატისტიკური მეთოდები (ძირითადად ნდობის ინტერვალებისა და ჰიპოთეზათა შემოწმების მეთოდები) სისტემაზე წინასწარი მონაცემების გათვალისწინებით. $e_0 = 0$ მომენტში კვანძი იმყოფება არადაკავებულ მდგომარეობაში. პირველი მოთხოვნის დრო განისაზღვრება F_A განაწილებიდან A_1 მნიშვნელობის გათამაშებით, გენერირებით (გენერირების მეთოდები იხ. თავი 3-ში) და მის 0-თან მიმატებით. ამის შემდეგ მდს გადადის შემდეგი (აქ პირველი) ხდომილების წარმოქმნის დროზე $e_1 = t_1$ (ნახ. 1.2-ზე ამოზნექილი ისრები წარმოადგენენ მდს-ს მიმდინარეობას). რადგან t_1 მომენტამდე სისტემა იმყოფება არა დაკავებულ მდგომარეობაში, იგი მყისიერად იწყებს მომსახურეობას რიგში მოთხოვნებზე $D_1 = 0$ დაკავებით. მისი მდგომარეობა იცვლება დაკავებულით. მოსულ მოთხოვნაზე მომსახურეობის დასრულების c_1 დრო განისაზღვრება მომსახურეობის S_1 დროის (F_S - განაწილებიდან გენერირებით) და მისი t_1 -ზე მიმატებით ($c_1 = t_1 + D_1 + S_1$). მეორე მოთხოვნის t_2 დრო გამოითვლება $t_2 = t_1 + A_2$ ფორმულით, სადაც A_2 გენერირდება F_A -დან. თუ $t_2 < c_1$ (როგორც ეს ნახ. 1.2-ზეა ნაჩვენები), მაშინ მდს-ს ჩვენება გადადის e_1 -დან $e_2 = t_2$ -შემდეგი ხდომილების დროზე (თუ $t_2 \geq c_1$, მაშინ მდს-ს ჩვენება გადავიდოდა e_1 -დან c_1 -ზე: $e_2 = c_1$). რადგან t_2 მომენტში მოსული მოთხოვნა აღმოაჩენს, რომ კვანძი უკვე დაკავებულია, მაშინ რიგში მყოფ მოთხოვნათა რიცხვი იზრდება 0-დან 1-მდე, ხოლო მოთოვნის დადგომის t_2 -დრო დაიმახსოვრება, ოღონდ ამ შემთხვევაში არ გენერირდება S_2 -მომსახურეობის დრო. მესამე მოთხოვნის დრო გამოითვლება $t_3 = t_2 + A_3$ ფორმულით, სადაც A_3 ხდომილება კვლავ გენერირდება F_A -დან. თუ $t_3 \geq c_1$ (როგორც ეს ნახაზზეა ნაჩვენები), მაშინ მდს-ს ჩვენება e_2 -დან გადადის $e_3 = c_1$ ჩვენებაზე. მაშინ მოთხოვნა, რომელზეც მომსახურეობა დასრულებულია, გადის სისტემიდან და იწყება რიგში მდგომი პირველი მოთხოვნის მომსახურეობა (მისი მოთხოვნის დროა t_2). გამოითვლება მისი რიგში დაკავების დრო $D_2 = c_1 - t_2$. ხოლო მისი გასვლის დროა $c_2 = c_1 + S_2$ (S_2 გენერირდება F_S -დან), ხოლო რიგში მყოფ მოთხოვნათა რიცხვი 1-დან 0-მდე მცირდება. თუ $t_3 < c_2$ მაშინ მდს-ს ჩვენება e_3 -დან გადადის $e_4 = t_3$ -შემდეგი ხდომილების წარმოქმნის დროზე და ა.შ. ასე გრძელდება მოდელირება. მოდელირება შეიძლება შეწყდეს (მაგალითად), როდესაც რიგში დაკავებულ მოთხოვნათა რიცხვი მიაღწევს წინასწარ განსაზღვრულ სიდიდეს ან სხვა მიზეზითაც.



ნახ.1.2. მდს-ს მიმდინარეობის მექანიზმი

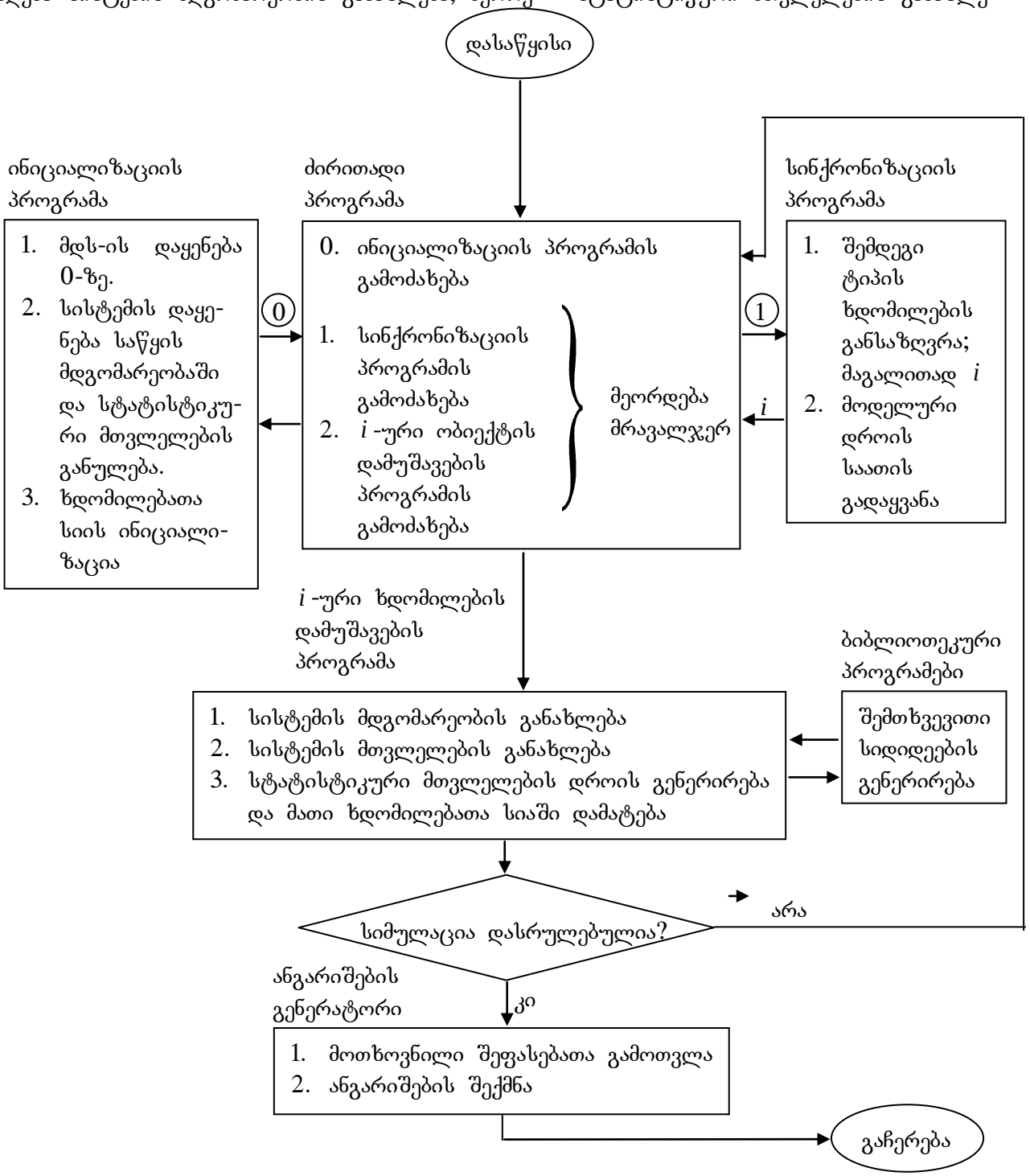
§1.2.2. დისკრეტულ-ხდომილებითი სიმულაციური მოდელის კომპონენტები და მისი ორგანიზაცია

თუმცა მოდელირება გამოიყენება სხვადასხვა რეალური სისტემების ანალიზისთვის, ამასთან ყველა დისკრეტულ-ხდომილებითი სიმულაციურ მოდელებს გააჩნიათ *საერთო კომპონენტები*. ამ კომპონენტთა *ლოგიკური ორგანიზაცია* უზრუნველყოფს *მოდელირების პროცესის დაპროგრამებას*, მის *გამართვას* და პროგრამის შემდგომ შესაძლო *ცვლილებებს*. დისკრეტულ-ხდომილებითი სიმულაციური მოდელი, რომელიც იყენებს ხდომილებიდან გადასვლის მდს-მექანიზმს და დაწერილია რომელიმე უნივერსალურ დაპროგრამების ენაზე, შეიცავს შემდეგ კომპონენტებს:

- *სისტემის მდგომარეობა* – მდგომარეობათა ცვლადების ერთობლიობა, აუცილებელი სისტემის აღწერისათვის დროის განსაზღვრულ მომენტში;
- *მოდელური დროის საათი* – ცვლადი, რომელიც უჩვენებს მოდელური დროის მიმდინარე მომენტს;
- *ხდომილებათა სია* – სია, რომელიც შეიცავს ყოველი შემდეგი განსხვავებული ტიპის ხდომილების დადგომის დროს;
- *სტატისტიკური მთვლელები* – ცვლადები, რომლებშიც ინახება სისტემის მახასიათებლებზე სტატისტიკური ინფორმაცია;
- *ინიციალიზაციის პროგრამა* – ქვეპროგრამა, რომელსაც სიმულაციური მოდელი მოყავს საწყის, ნულოვან მდგომარეობაში;
- *სინქრონიზაციის პროგრამა* – ქვეპროგრამა, რომელიც ხდომილებათა სიაში ეძებს შემდეგ ხდომილებას და მდს-ს ჩვენება გადაყავს ახალი ხდომილების დადგომის დროზე.
- *ხდომილებათა დამუშავების პროგრამა* – ქვეპროგრამა, რომელიც ანახლებს სისტემის მდგომარეობას, როდესაც დგება განსაზღვრული ტიპის ხდომილება (ყოველი ტიპის ხდომილებისთვის არსებობს ხდომილების დამუშავების ცალკე ქვეპროგრამა);
- *ბიბლიოთეკური პროგრამა* – ქვეპროგრამათა კრებული, რომელიც გამოიყენება შემთხვევითი დაკვირვებათა გენერაციისათვის (რომლებიც გამოიყენებიან სიმულაციურ მოდელში).
- *ანგარიშების გენერატორი* – ქვეპროგრამა, რომელიც ითვლის სატატისტიკური მთვლელებიდან სისტემის მუშაობის კრიტერიუმების შეფასებებს და იძლევა მოდელირების დასრულების ანგარიშს.
- *ძირითადი პროგრამა* – ქვეპროგრამა, რომელიც იძახებს სინქრონიზაციის პროგრამას, რათა განისაზღვროს შემდეგი ხდომილება. შემდეგ მართვას გადასცემს შესაბამისი ხდომილების დამუშავების პროგრამას, რათა მოხდეს სისტემის მდგომარეობის განახლება. ძირითად პროგრამას ასევე შეუძლია შეასრულოს მოდელირების შეწყვეტის კონტროლი და გამოიძახოს მოდელირების დასრულების ანგარიშის გენერატორიც.

ზემოთ ჩამოთვლილ კომპონენტებს შორის ლოგიკური მიმართებები ნაჩვენებია ნახ. 1.3-ზე. მოდელირება იწყება 0-მომენტში. ძირითადი პროგრამა გამოიძახებს ინიციალიზაციის პროგრამას, რომელიც მდს-ის ჩვენებას დააყენებს 0-ვან მომენტისთვის. შემდეგ სისტემას დააყენებს საწყის მდგომარეობაში, სტატისტიკურ მთვლელებს მოიყვანს საწყის მდგომარეობაში და ინიციალიზაციას გაუკეთებს ხდომილებათა სიას. მართვის ძირითად პროგრამაზე დაბრუნების შემდეგ გამოიძახება სინქრონიზაციის პროგრამა, რათა განისაზღვროს უახლოესი ხდომილების ტიპი. თუ ის არის i-ტიპი, მაშინ სიმულაციის დროის საათი გადადის i-ური ტიპის ხდომილების დადგომის დროზე და მართვა უბრუნდება ძირითად პროგრამას. ძირითადი პროგრამა ააქტიურებს i-ური ტიპის ხდომილების დამუშავების პროგრამას, ამ დროს ხდება სამი ტიპის მოქმედება: პირველი – i-ური ტიპის ხდომილების დადგომასთან დაკავშირებით

ხდება სისტემის მდგომარეობის განახლება; მეორე – სტატისტიკური მთვლელების განახლე-



ნახ. 1.3

ბის გზით გროვდება ინფორმაცია სისტემის მუშაობის შეფასების კრიტერიუმებზე; მესამე- გენერირდება მომავალი ხდომილებების წარმოქმნის დრო და ინფორმაცია ამის შესახებ ჩაემატება ხდომილებათა სიას. ხშირად, მომავალი ხდომილებათა წარმოქმნის დროის განსაზღვრისას, წარმოიქმნება შემთხვევითი სიმულაციის, გათამაშების ანუ გენერირების აუცილებლობა. ასეთ შემთხვევით დაკვირვებას *შემთხვევითი სიდიდე* ეწოდება. i -ური ხდომილების დამუშავების პროგრამის დამთავრების შემდეგ ძირითად პროგრამაში სრულდება შემოწმება იმ მიზნით, რომ გაირკვეს – შეწყდეს მოდელირება თუ არა? თუ მოდელირება წყდება, მაშინ ძირითადი პროგრამიდან გამოიძახება ანგარიშების გენერატორი, რათა გამოითვალოს სისტემის მუშაობის შეფასების კრიტერიუმები და შეიქმნას ანგარიში. თუ მოდელირების

დასრულების დრო ჯერ არ დამდგარა, მაშინ მართვა კვლავ გადაეცემა ძირითად პროგრამას და ციკლი "ძირითადი პროგრამა ↔ სინქრონიზაციის პროგრამა ↔ ძირითადი პროგრამა ↔ ხდომილებათა დამუშავების პროგრამა ↔ სიმულაციის დასრულების პირობის შემოწმება" მეორდება მანამ, სანამ არ შესრულდება გაჩერების პირობა.

ცოტა რამ სისტემის მდგომარეობებზე. როგორც უკვე ადრე ითქვა სისტემა არის გარკვეული *ობიექტების* ერთობლიობა. ობიექტები აღიწერებიან მნიშვნელობებით, რომელთაც *ატრიბუტები* ეწოდებათ. დისკრეტულ-ხდომილებითი სიმულაციურ მოდელში ეს ატრიბუტები წარმოადგენს სისტემის მდგომარეობის ნაწილს. ობიექტები, რომელთაც გააჩნიათ საერთო თვისებები ერთიანდებიან *სიებში* (ფაილებში, ნაკრებებში). ყოველი ობიექტისთვის არსებობს *ჩანაწერი* სიაში, რომელიც ობიექტის ატრიბუტებისგან შედგება. ობიექტების სიაში განლაგების საკითხს წყვეტს გარკვეული *წესი* (განლაგების ეფექტური წესები ცნობილია მონაცემთა სტრუქტურებში). ზემოთ აღწერილი სისტემისთვის ობიექტებს წარმოადგენს მომსახურეობის კვანძი და მოთხოვნები შეკვეთებზე. აქ ობიქტი-კვანძს გააჩნია ატრიბუტი "დაკავებულია თუ თავისფალია". მოთხოვნებს, რომლებიც ჩაყენებულია რიგში, გააჩნიათ ატრიბუტი-"დადგომის დრო". (რიგში დაკავებულ მოთხოვნათა რიცხვი ასევე შეიძლება ჩაითვალოს ობიექტ-კვანძის ატრიბუტად). როგორც მე-4-ე თავში გამოჩნდება მოთხოვნები ერთიანდებიან სიებში.

დისკრეტულ ხდომილებითი მოდელების პროგრამების ორგანიზაცია და ფუნქციონირება, რომელშიც გამოიყენება ხდომილებიდან ხდომილებაზე გადასვლის მექანიზმი, საკმაოდ ტიპიურია, თუ პროგრამა დაწერილია C უნივერსალურ ენაზე. სიმულაციაზე ასეთ მიდგომას *ხდომილების დაგეგმვა* ეწოდება, რადგანაც მომავალი ხდომილების დრო მოდელში ცხადად მიეთითება და იგეგმება მოდელის მომავალში.

§ 1.3. უწყვეტი მოდელირების შესახებ

უწყვეტი მოდელირება – სისტემის დროში მოდელირებაა, რომლის წარმოდგენაში მდგომარეობის ცვლადები იცვლებიან *უწყვეტად* დროსთან მიმართებაში. როგორც ყოველთვის *უწყვეტ სიმულაციურ მოდელებში* (უსმ) გამოიყენება *დიფერენციალური განტოლებები*, რომლებიც ადგენენ მიმართებებს დროში სისტემის ცვლადების ცვლილების სიჩქარეებისთვის. თუ დიფერენციალური განტოლებანი ძალიან მარტივია, ისინი შეიძლება ამოიხსნას ანალიზურად, რათა წარმოდგეს მდგომარეობის ცვლადების მნიშვნელობანი დამოკიდებული დროზე 0-მომენტიდან. "დიდი" უწყვეტი მოდელებისთვის ანალიზური ამონახსნები არ არსებობს, ჩაწერილი ცხადი სახით, რომელთა გამოთვლაა შესაძლებელი, მაგრამ არსებობს მათემატიკური აპარატი: დიფერენციალური განტოლებათა რიცხვითი ინტეგრირება-რიცხვითი ანალიზის ნაწილი, რომლის გამოყენებით უმეტეს წილად შესაძლებელია განტოლებათა *მიახლოებითი რიცხვითი ამონახსნის მიღება.* და რომლის გამოყენება მისაღებია სიმულაციური უწყვეტი მოდელირების ამოცანებში.

სიმულაციური მოდელირების რამდენიმე პროგრამული უზრუნველყოფა (SIMULINK, Dymola, Arena, AweSim, Extend და სხვა) აკეთებს უსმ-ს მაღალ სამომხმარებლო დონეზე (უწყვეტი სიმულაციური მოდელირების უკეთ გაცნობის მიზნით მიმართეთ ჟურნალს სახელწოდებით *Simulation*).

მაგალითი 1.2. განვიხილოთ უწყვეტი მოდელი, რომელიც აღწერს ორ პოპულაციას შორის მეტოქეობას. ამ ტიპის ბიოლოგიური მოდელები გამოირჩევიან "მტაცებელი-მსხვერპლი"-ს მოდელის სახელწოდებით (პირველად განიხილებოდა, Braun, 1975 და Gordon, 1978-ავტორების მიერ).

გარემო წარმოდგენილია ორი პოპულაციით: მტაცებლები და მსხვერპლი, რომლებიც ურთიერთქმედებენ ერთმანეთთან. მსხვერპლი პასიურია, ხოლო მტაცებლები დამოკიდებულია მსხვერპლის პოპულაციაზე, რადგან იგი წარმოადგენს მტაცებლის ერთადერთ საკვებს (მაგ.

მტაცებლები შეიძლება იყოს ზვიგენები, ხოლო მსხვერპლი თევზები, რომლითაც იკვებება ზვიგენი).

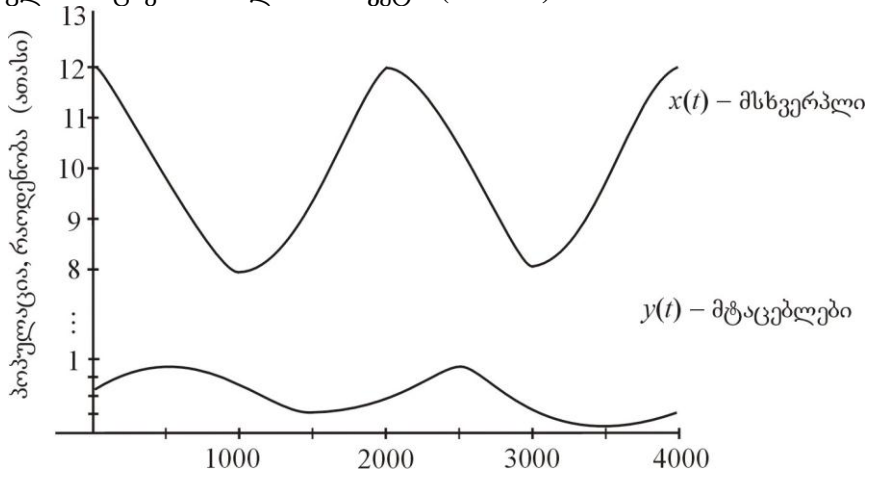
ვთქვათ $x(t)$ და $y(t)$ აღნიშნავს შესაბამისად მსხვერპლის და მტაცებელთა პოპულაციების რაოდენობებს t -დროის მომენტში. დავუშვათ, რომ მსხვერპლის პოპულაციას აქვს კვების შეუზღუდავი მარაგი და მტაცებელთა არ არსებობის შემთვევაში მისი ზრდის ტემპი შეადგენს $r \cdot x(t)$ სიდიდეს, რაიმე $r > 0$ რიცხვისთვის (r -არის სხვაობა ბენებრივად დაბადებისა და ბუნებრივად დაღუპვის დონეებს შორის). მსხვერპლს და მტაცებელს შორის ურთიერთდამოკიდებულების არსებობა გვაძლევს საფუძველს დავუშვათ, რომ მსხვერპლის დაღუპვის ტემპი პირდაპირპროპორციულია ორივე პოპულაციათა რაოდენობების $x(t) \cdot y(t)$ ნამრავლის. ამიტომ $\frac{dx}{dt}$ -მსხვერპლის პოპულაციის ცვლილების საერთო ტემპი შეიძლება წარმოვიდგინოთ როგორც სხვაობა:

$$\frac{dx}{dt} = rx(t) - ax(t)y(t), \tag{1.1}$$

სადაც $a > 0$ -პროპორციის კოეფიციენტია. რადგან თვით მტაცებელთა არსებობა დამოკიდებულია მსხვერპლის პოპულაციაზე, ამიტომ მსხვერპლის მოპოვების გარეშე მტაცებელთა პოპულაციის შემცირების ტემპი შეადგენს $sy(t)$ -ს, რაიმე $s > 0$ რიცხვისთვის. გარდა ამისა ორი პოპულაციის ურთიერთქმედება იწვევს მტაცებელთა პოპულაციის ზრდას, რომლის ტემპი პირდაპირპროპორციულია $x(t)y(t)$ -ნამრავლის. $\frac{dy}{dt}$ -მტაცებელთა პოპულაციის ცვლილების საერთო ტემპი შეადგენს:

$$\frac{dy}{dt} = -sy(t) + bx(t)y(t), \tag{1.2}$$

სადაც $b > 0$ პროპორციულობის კოეფიციენტია. პოპულაციათა სასტარტო მნიშვნელობებია: $x_0 = x(0) > 0$ და $y_0 = y(0) > 0$. $t > 0$ მნიშვნელობებისთვის (1.1)-(1.2) განტოლებათა სისტემას გააჩნია ერთადერთი ამონახსნი (Braun, 1975), თანაც $x(t) > 0$ და $y(t) > 0$. მაშასადამე ურთიერთქმედებაში არცერთი პოპულაცია არ გაწყდება. ამასთან $\{x(t); y(t)\}_{t \geq 0}$ - ამონახსნი პერიოდულია რაიმე $T > 0$ უმცირესი დადებითი პერიოდით: $x(t+nT) = x(t)$, $y(t+nT) = y(t)$, $n \in Z$. ეს შედეგი არაა მოულოდნელი, რადგან მტაცებელთა პოპულაციის ზრდით მცირდება მსხვერპლის პოპულაცია. ეს შემდეგში მსხვერპლის პოპულაციის შემცირებას იწვევს. ხოლო ეს თავისთავად იწვევს მტაცებელთა პოპულაციის შემცირებას და ა.შ. პერიოდულად. $r = 0.01$; $a = 2 \cdot 10^{-6}$; $s = 0.01$; $b = 10^{-6}$; $x(0) = 12000$; $y(0) = 600$ შემავალი მონაცემებისათვის მიღებულია (1.1), (1.2) ამოცანის რიცხვითი ამონახსნი. გამოყენებულია რიცხვითი ანალიზის პაკეტი (ნახ. 1.4).



ნახ. 1.4

მოყვანილი მაგალითი დეტერმინისტულია და მოდელი არ შეიცავს შემთხვევით კომპონენტებს. თუმცა პარამეტრები r, a, s და b-შეიძლება იყვნენ დროზე დამოკიდებული და თანაც შემთხვევითიც. ასეთი უწყვეტი, თუნდაც მარტივად გასაგები მოდელები ძალიან რთულდება და მოითხოვს *სტოქასტური სიმულაციური მოდელირების მაღალ დონეს.*

“ნებისმიერი, ვინც ეჭვის თვალით უყურებს შემთხვევითი რიცხვების მიღების არითმეტიკულ მეთოდებს, სცოდავს ყოველგვარი ეჭვის გარეშე”

ჯონ ფონ ნეიმანი (1951)

თავი 2. შემთხვევითი რიცხვების გენერატორები

შესავალი

ნებისმიერი სისტემის ან პროცესის სიმულაციური მოდელირება, რომელიც შეიცავს შემთხვევით კომპონენტებს, ითვალისწინებს შემთხვევითი რიცხვების გენერირების (სიმულაციის, გათამაშების) მეთოდების გამოყენებას. მაგალითად: მასობრივი მომსახურეობის სისტემების მოდელებისთვის მოითხოვება მომდევნო შეკვეთებს შორის დროის ინტერვალის, მომსახურეობის დროის ინტერვალის და ა.შ. პარამეტრთა მნიშვნელობები. მათ მისაღებად გამოიყენებოდა კონკრეტული განაწილებანი (მაგ., ექსპონენციალური).

ამ და შემდეგ თავში ჩვენ გავეცნობით, თუ როგორ შეიძლება საჭირო ალბათური განაწილებიდან მივიღოთ შემთხვევითი მნიშვნელობათა სიმულაცია (გენერირება, გათამაშება), რათა ეს მნიშვნელობები გამოყენებული იქნას სიმულაციურ მოდელირებაში. აქ სიმულაცია, გენერირება, გათამაშება – სინონიმებია. ისინი გულისხმობენ ხელოვნურად, ანალიზური მეთოდებითა თუ ალგორითმებით, ცხრილებით და ა.შ. მივიღოთ რიცხვითი მიმდევრობები, რომელთაც ისეთივე თვისებები ექნებათ, როგორც შემთხვევით რიცხვთა მიმდევრობებს (ამის შესახებ უფრო ვრცლად ქვემოთ ვისაუბრებთ).

რეალურად, ეს თავი დაეთმობა [0,1]-ზე თანაბრად განაწილებული γ შემთხვევითი სიდიდის გენერირების, სიმულაციის მეთოდებს. ამ სიდიდეს შემთხვევითი რიცხვი ეწოდება, ასეთი სიდიდე $U(0,1)$ სიმბოლოთი აღინიშნება. როგორც ცნობილია (თავი 3), $U(0,1)$ კანონით გენერირებული შემთხვევითი, დამოუკიდებელი რიცხვითი მიმდევრობიდან შეიძლება მიღებული იქნას (გენერირდეს) ნებისმიერი სხვა განაწილების მქონე შემთხვევითი სიდიდეები და პროცესები. ამიტომ მაღალი ხარისხის შემთხვევითი რიცხვების გენერატორების აგება და ანალიზი მნიშვნელოვანი ამოცანაა მოდელირებასა და სიმულაციებში.

შემთხვევითი რიცხვების გენერირების მეთოდოლოგიას გააჩნია ხანგრძლივი და საინტერესო ისტორია. ყველაზე ადრეული გენერირების მეთოდები იყენებდნენ ფიზიკურ ხელსაწყოებს და შემთხვევითი რიცხვები ხელით მიიღებოდა. მაგ.: სათამაშო კამათლის აგდება, ბანქოს დარიგება ან კიდევ წინასწარ დანომრილი ბურთულების ამოღება ყუთიდან და ა.შ. დღეს-დღეობით მრავალი სათამაშო ლატარეა ანალოგიურად ფუნქციონირებს. XX საუკუნის დასაწყისში უკვე ყურადღება მიექცა უფრო სწრაფი გენერატორების კონსტრუქციებს. 1930-იან წლებში სტატისტიკოსებმა კენდალმა და ბაბინქტონ-სმიტმა გამოიყენეს სწრაფად მბრუნავი დისკი 100 000 შემთხვევითი ციფრის მისაღებად. უკვე მოგვიანებით შეიქმნა ელექტრონული სქემები, რომლებიც შემთხვევითი რიცხვების გენერირებისათვის იყენებდნენ პულსაციურ ელექტრონულ “ლამპებს”, რომლებიც 1 წმ-ში იძლეოდნენ დაახლოებით 50 შემთხვევითი რიცხვს. ერთ-ერთი ასეთი მოწყობილობა იყო – Electronic Random Number Indicator Equipment (ERNIE) [1], რომელიც 1960-იან წლებში ბრიტანეთის საფოსტო უწყების მიერ გამოყენებული იქნა ლატარეაში გამარჯვებულთა გამოსავლენად.

კომპიუტერის განვითარებისა და გავრცელების შედეგად სულ უფრო მეტი ყურადღება ეთმობოდა ისეთი გენერატორების შექმნას, რომლებიც შესაბამისობაში იყო კომპიუტერის მონაცემებთან და მის მუშაობასთან. ერთ-ერთი ასეთი საშუალება იყო ERNIE-ს ტიპის გენერატორების პირდაპირი მიერთება კომპიუტერებთან, როგორც გარე მოწყობილობა. ასეთი

გენერატორების უარყოფითი მხარეებია: ა) გენერირებული მიმდევრობის პრაქტიკულად გამეორების შეუძლებლობა, ბ) გენერირებული შემთხვევითი რიცხვების კომპიუტერულ პროგრამებთან ინტერფეისში გადაცემის საკმაოდ დაბალი სიჩქარე. ამ მეთოდების ალტერნატიულ მეთოდს წარმოადგენს დიდი მოცულობის მქონე წინასწარ მომზადებული შემთხვევითი რიცხვების (რომლებიც აკუმულირებულია მეხსიერების გარე მატარებლებზე) მასივების გამოყენება კომპიუტერული პროგრამებისათვის (შემთხვევითი რიცხვების ცხრილები). ცხრილების უარყოფითი მხარეები იყო შედარებით ნელი პროცესი – რიცხვების მეხსიერების გარე მატარებლებიდან ოპერაციულ მეხსიერებაში გადაგზავნა (თუმცა დღეს-დღეობით მოდელირების პროექტებში გამოიყენება დაახლოებით არაუმეტეს 10^6 შემთხვევითი რიცხვი და მათი გამოყენების ეფექტურობა მეხსიერების გარე მატარებლებიდან საკმაოდ მაღალია). ეს პრობლება იყო სწორედ ის, რომ 1940–1950-იან წლებში მკვლევარები ჩაერთვნენ ე.წ. *რიცხვითი, არითმეტიკული შემთხვევითი რიცხვების გენერატორების კონსტრუირებაში.* ეს მეთოდები წარმოადგენს მიმდევრობით (უფრო ხშირად რეკურენტულ) ალგორითმებს, სადაც ყოველი ახალი რიცხვი განისაზღვრება წინა ერთი ან რამოდენიმე შემთხვევითი რიცხვით, განსაზღვრული წესით წარმოდგენილ ფორმულაში მათი ჩასმით ან ალგორითმში შეტანით.

პირველი ასეთი არითმეტიკული გენერატორი ფ. ნეიმანისა და ვ. მეტროპოლისის მიერ (1940-იანი წლები) იქნა აგებული "შუას კვადრატის" მეთოდის სახელწოდებით. გთავაზობთ ამ მეთოდის გამოყენების მაგალითს: z_0 – ოთხნიშნა მთელი რიცხვი ავიყვანოთ კვადრატში. მივიღებთ რვანიშნა მთელ რიცხვს (თუ საჭირო შეიქმნა მარცხნივ დავუმატებთ იმდენ ნულს, რომ კვადრატი შევავსოთ 8 ნიშნამდე). z_1 – შემთხვევითი ოთხნიშნა მთელი რიცხვის როლში ავიღოთ მიღებული კვადრატის – 8 ნიშნა რიცხვის – შუა 4 ციფრისაგან შედგენილი რიცხვი, რომელსაც მარცხენა ციფრის წინ დაეწერება მძიმე და მის წინ კი 0. ასე მიიღება γ_1 – პირველი შემთხვევითი რიცხვი. ეს პროცესი რეკურენტულად გაგრძელდება. მივიღებთ $\gamma_1, \gamma_2, \ldots$ შემთხვევით რიცხვთა მიმდევრობას.

შემთხვევითი რიცხვების ცხრილი, გენერირებული "შუას კვადრატის" მეთოდით
($z_0 = 7182$)

i	z_i	γ_i	z_i^2
0	7182	–	51 581 124
1	5811	0,5811	33 767 721
2	7677	0,7677	58 936 329
3	9363	0,9363	87 665 769
4	6657	0,6657	44 315 649
5	3156	0,3156	09 960 336
⋮	⋮	⋮	⋮

შეიძლება გვეჩვენოს, რომ "შუას კვადრატის" მეთოდი უზრუნველყოფს კარგი შემთხვევითი რიცხვების მიმდევრობის მიღებას, მაგრამ ეს ასე არ არის. ერთ-ერთი უარყოფითი თვისება ის არის, რომ გენერატორს გააჩნია მყარი თვისება მოგვცეს რიცხვები, რომლებიც 0-სკენ ისწრაფვიან, რაც თანაბარ განაწილებას არღვევს.

ეს მეთოდი სულაც არაა შემთხვევითი, ანუ წინასწარ ამოუცნობი. სინამდვილეში, თუ ჩვენ ვიცით საწყისი z_0 რიცხვი, მაშინ $\{\gamma_i\}_{i\geq 1}$ გენერირებული მიმდევრობა წინასწარ განსაზღვრულია. ეს *შენიშვნა* ეხება *ნებისმიერ არითმეტიკულ გენერატორს.* ასე ჩვენ შეიძლება მივიდეთ მისტიკურ მსჯელობამდე ნამდვილი შემთხვევითი რიცხვების ბუნებასთან მიმართებაში (ხშირად არითმეტიკულ გენერატორებს ეწოდებათ *ფსევდოშემთხვევითი* და მათი საშუალებით მიღებულ რიცხვებს კი ფსევდოშემთხვევითი რიცხვები [6], რა თქმა უნდა ეს გან-

მარტება უფრო შეესაბამება რეალობას, თუმცა ჩვენ მას მომავალში არ გამოვიყენებთ, მაგრამ ვიგულისხმებთ).

ხშირად ციტირებას უკეთებენ ფ. ნეიმანის შემდეგ შენიშვნას [2]:

"... ნებისმიერი, რომელიც განიხილავს შემთხვევითი რიცხვების მიღების არითმეტიკულ მეთოდებს, ცოდვებში ვარდება. რადგანაც უკვე მრავალჯერ ითქვა, რომ არ არსებობს შემთხვევითი რიცხვი, მაგრამ არსებობენ შემთხვევითი რიცხვების მიღების მეთოდები და მკაცრი არითმეტიკული პროცედურა, რა თქმა უნდა, ასეთ მეთოდებს არ განეკუთვნება... აქ ჩვენ საქმე გვაქვს შემთხვევითი ციფრების მიღების ჩვეულებრივ "კულინარიულ რეცეპტებთან ...".

თუმცა საკმაოდ იშვიათად, მაგრამ მაინც აღნიშნავენ იგივე ფ. ნეიმანის ნათქვამს, რომ ეს ე.წ. "კულინარიული რეცეპტები" "... ალბათ არ შეიძლება იქნას გამართლებული, თუმცა უბრალოდ აუცილებლად უნდა იქნეს შეფასებული მათი რეზულტატები. აღნიშნული რეცეპტებით გენერირებული ციფრების სტატისტიკური გამოკვლევა უნდა სრულდებოდეს, ხოლო ყოველმხრივი, საფუძვლიანი კვლევა ამ შემთხვევაში უსარგებლოა. თუ გენერირებული ციფრები კარგად გამოიყენებიან ერთი ამოცანისათვის, მაშინ ისინი როგორც წესი ასევე წარმატებით გამოიყენება იგივე ტიპის სხვა ამოცანების ამოხსნებშიც ..." [1].

ჩვენ ვეთანხმებით უმრავლესობა ავტორებს იმაში, რომ არითმეტიკულ გენერატორებს, რომლებიც სათანადოდ კონსტრუირებულია, შეუძლიათ ისეთი რიცხვების გენერირება, რომლებიც წარმოადგენენ $U(0,1)$ კანონით განაწილების დამოუკიდებელ შერჩევებს, რადგან ასეთი გენერატორები გადიან მთელ შემოწმების რიგ სტატისტიკურ თუ თეორიულ ტესტებს.

შემთხვევითი რიცხვების კარგ არითმეტიკულ გენერატორს უნდა გააჩნდეს შემდეგი თვისებები [2]:

1. მიღებული რიცხვები თანაბრად უნდა იყვნენ განაწილებული [0,1]-ში და ერთმანეთში არ უნდა ჰქონდეთ კორელაცია;
2. გენერატორი უნდა გამოირჩეოდეს სწრაფქმედებით და არ უნდა ითხოვდეს მეხსიერების დიდ რესურსებს;
3. გენერატორს უნდა გააჩნდეს უნარი დამოუკიდებელი გენერირებული ნაკადების (იგივე სერიების) სიმულირებისა, რათა შესრულდეს მაღალი ხარისხის მოდელირება;
4. გენერატორს უნდა გააჩნდეს მომდევნო ნაკადის მიღების საშუალება, დაწყებული წინა ნაკადის ბოლო ელემენტიდან. უმრავლესობა თანამედროვე გენერატორებში გათვალისწინებულია მაღალი სწრაფქმედება და 2 ან მეტი პარალელური დამოუკიდებელი ნაკადის გენერირება;
5. გენერატორები უნდა უზრუნველყოფდნენ გენერირებული შერჩევების დამოუკიდებლობას და თანაბრად განაწილებას (ასრულებენ შესაბამის სტატისტიკურ ჰიპოთეზებს მაღალი სანდოობით).

შევნიშნოთ, რომ შემთხვევითი რიცხვების გენერატორების კონსტრუირების თემა საკმაოდ რთულია. ის მოიცავს ისეთ დისციპლინებს, როგორიცაა აბსტრაქტული ალგებრა და რიცხვთა თეორია, ერთის მხრივ, და მეორეს მხრივ, სისტემური დაპროგრამება და კომპიუტერის არქიტექტურა.

2.1. წრფივი კონგრუენტული გენერატორები

მოდელირებასა და სიმულაციაში დღეს ხშირად გამოიყენება წრფივი კონგრუენტული გენერატორები (წკგ), რომლის ავტორია ლემერი (Lehmer, 1951). მასში $z_1, z_2, \ldots$ – მთელი რიცხვების მიმდევრობა განისაზღვრება

$$z_i = (az_{i-1} + c) (\mathrm{mod}\ m), \qquad i = 1,2,\ldots, \tag{2.1}$$

რეკურენტული ფორმულით, სადაც m (მოდული), c (წანაცვლება), a (მამრავლი) და z_0 (საწყისი რიცხვი ან მნიშვნელობა) პარამეტრები არაუარყოფითი მთელი რიცხვებია. (2.1)

ფორმულის თანახმად $az_{i-1}+c$ უნდა გაიყოს m-ზე და z_i-ს მიენიჭოს *განაყოფით მიღებული ნაშთი*. ამიტომ $0 \le z_i \le m-1$. იმისათვის, რომ მივიღოთ $U(0,1)$ კანონით განაწილებული γ_i შემთხვევითი რიცხვი, საჭიროა z_i-ს ნორმირება m-ით:

$$\gamma_i = z_i/m, \qquad i = 1,2,\ldots. \tag{2.2}$$

(2.1) ფორმულაში მთავარი კონცენტრაცია რა თქმა უნდა z_i-ის გამოთვლას დაეთმობა, თუმცა მხედველობაში უნდა მივიღოთ (2.2) ფორმულით გაყოფის ოპერაციის ზუსტი ხასიათი, რადგან განსხვავებულ კომპიუტერსა და რიცხვების კომპილიატორებს გააჩნიათ განსხვავებული საშუალებები. პარამეტრების დამატებითი შეზღუდვებია

$$m > 0, \qquad m > a, \qquad m > c, \qquad m > z_0. \tag{2.3}$$

მათემატიკური ინდუქციით შეიძლება ვაჩვენოთ, რომ

$$z_i = \left[a^i z_0 + \frac{c(a^i - 1)}{a-1} \right] (\text{mod } m), \qquad i = 1, 2,\ldots \qquad (a \ne 1), \tag{2.4}$$

ანუ z_i-ის ყოველი მნიშვნელობა სრულად განისაზღვრება m, a, c და z_0 პარამეტრებით, რაც კიდევ ერთხელ მიუთითებს (2.1)–(2.3) წრფივი კონგრუენტული გენერატორის არაშემთხვევითობაზე და მის დეტერმინისტულ ხასიათზე. თუმცა აღნიშნული პარამეტრების *სათანადოდ შერჩევით* მომავალში შევეცდებით მივიღოთ z_i მიმდევრობის ისეთი განსაზღვრება, როდესაც შესაბამისი γ_i სიდიდეები *წარმოდგებიან როგორც* დამოუკიდებელი და თანაბრად განაწილებული შემთხვევითი სიდიდეები, განაწილებული $U(0,1)$ კანონით. ამ დროს ეს სიდიდეები მოწმდებიან მთელი რიგი ტესტებით (იხ. §2.5).

შევნიშნოთ, რომ წკგ-ის გამოყენებით γ_i სიდიდეები ღებულობენ მხოლოდ შემდეგ რაციონალურ მნიშვნელობებს:

$$0, \quad \frac{1}{m}, \quad \frac{2}{m}, \ldots, \frac{m-1}{m}. \tag{2.5}$$

ასე რომ ჩვენ მოგვიწევს m მოდულის მნიშვნელობად რაც შეიძლება დიდი მთელი რიცხვის აღება; მაგ. 10^9 და მეტი, რომ (2.5)-ის ტიპის რიცხვები რაც შეიძლება მეტი მკვრივი იყოს $(0,1)$ შუალედში. $m \ge 10^9$ მოდულისთვის არსებობს სულ მცირე 1 მილიარდი შესაძლო მნიშვნელობა. დიდი m-ის აღებით ჩვენ უზრუნველვყოფთ შედარებით კარგ მიახლოებას $U(0,1)$ რეალურ უწყვეტ განაწილებასთან. როგორც მონაჰანი (Monahan, 1985) აღნიშნავს, დისკრეტული წილადებით გამოთვლილი γ_i მნიშვნელობათა რაოდენობის შეზღუდვამ შეიძლება გამოიწვიოს სისტემების მოდელირების მნიშვნელოვანი ცდომილებანი.

განვიხილოთ მაგალითი წკგ-ს აგებაზე:

$$z_i = (5z_{i-1} + 3)(\text{mod } 16), \quad z_0 = 7$$

i	z_i	γ_i	i	z_i	γ_i	i	z_i	γ_i	i	z_i	γ_i
0	7	–	5	10	0.625	10	9	0.563	15	4	0.250
1	6	0.375	6	5	0.313	11	0	0.000	16	7	0.438
2	1	0.063	7	12	0.750	12	3	0.188	17	6	0.375
3	8	0.500	8	15	0.938	13	2	0.125	18	1	0.063
4	11	0.688	9	14	0.875	14	13	0.813	19	8	0.500

ცხრილში წარმოდგენილია z_i და γ_i-ის მნიშვნელობები (მძიმის შემდეგ 3 თანრიგის სიზუსტით) $i = 1,2,\ldots 19$ -სთვის. შევნიშნოთ, რომ $z_{17} = z_1$, $z_{18} = z_2$ და ა.შ. ანუ $i = 17, 18,\ldots,32$ ინდექსებისთვის ვღებულობთ იგივე მნიშვნელობებს, რომლებიც მიღებული

იყო $i = 1,2,\ldots 16$ ინდექსებისთვის იგივე წესით. გასაგებია, რომ პრაქტიკაში ამ გენერატორის გამოყენება არ შეიძლება, რადგან m საკმაოდ მცირეა.

აღნიშნულ გენერატორში *ციკლურობა* გარდაუვალია. და საერთოდ (2.1)–(2.3) ტიპის გენერატორებში როგორც კი z_i მიიღებს ისეთ მნიშვნელობას, რომელიც უკვე იყო გენერირებული, შემდეგ უკვე გენერირდება იგივე მიმდევრობა. ეს ციკლი მეორდება უსასრულოდ. ციკლის სიგრძეს *გენერატორის პერიოდი* ეწოდება. რადგან $0 \le z_i \le m-1$ და z_i დამოკიდებულია z_{i-1}-ზე, ცხადია, პერიოდის სიგრძე არ აღემატება m სიდიდეს.

თუ პერიოდის სიგრძე ემთხვევა m მოდულს, მაშინ იტყვიან, რომ წკგ-ს აქვს *სრული პერიოდი* (როგორც ეს წარმოდგენილ მაგალითშია).

თუ გენერატორს გააჩნია სრული პერიოდი, მაშინ ცხადია, რომ როგორიც არ უნდა ავიღოთ საწყისი z_0 მნიშვნელობა, $\{0,1,\ldots,m-1\}$ მიმდევრობა გადალაგდება გარკვეულ ინვერსიაში, რომელიც წარმოადგენს ციკლს. ზემოთ მოყვანილი მაგალითის ციკლია: 7, 6, 1, 8, 11, 10, 5, 12, 15, 14, 9, 0, 3, 2, 13, 4. თუ გენერატორის *პერიოდი ნაკლებია სრულზე*, მაშინ ციკლის სიგრძე დამოკიდებული იქნება z_0 საწყისი მნიშვნელობის არჩევაზე.

სიმულაციური მოდელირების ფართომასშტაბიან პროექტებში შეიძლება გამოყენებული იყოს ასეული რაოდენობის შემთხვევითი რიცხვები. ცხადია, ამისთვის საჭიროა დიდი პერიოდის მქონე გენერატორები. უფრო მეტიც, უკეთესია სრული პერიოდის მქონე გენერატორის გამოყენება, რადგან 0-დან $m-1$-მდე რიცხვები გენერირებისას მხოლოდ ერთხელ გამოითვლება ციკლში. ეს კი z_i რიცხვების თანაბარი განაწილების აუცილებელი პირობაა. ასე რომ მნიშვნელოვანია m, a და c პარამეტრებს შორის ისეთი დამოკიდებულება, რომელიც (2.1)–(2.3) გენერატორში უზრუნველყოფს სრულ პერიოდს.

თეორემა 2.1 (Hull and Dobell, 1962). (2.1)–(2.3) *ტიპის გენერატორს გააჩნია სრული პერიოდი მაშინ და მხოლოდ მაშინ, როცა*

ა) c *და* m *პარამეტრები ურთიერთ მარტივია (მათი უდიდესი საერთო გამყოფია 1);*

ბ) *თუ* m*-ს გააჩნია მარტივი გამყოფი* q*, მაშინ* $a-1$ *იყოფა* q*-ზე;*

გ) *თუ* m *იყოფა 4-ზე, მაშინ* $a-1$ *იყოფა 4-ზე.*

ზემოთ მოყვანილი მაგალითი აკმაყოფილებს ამ მოთხოვნებს:

ა) $c = 3$ და $m = 16$ ურთიერთ მარტივია;

ბ) $m = 16 = 2^4$*-ს გააჩნია მარტივი გამყოფი* $q = 2$*,* მაშინ $a-1 = 5-1 = 4$-იც იყოფა q-ზე;

გ) პირობა რა თქმა უნდა სრულდება.

გენერატორში *სრული პერიოდის* ან *დიდი სიგრძის პერიოდის მიღწევა* ერთ-ერთი *მნიშვნელოვანი ამოცანაა*. როგორც ადრე აღვნიშნეთ, საჭიროა კიდევ ისეთი თვისებების ქონა, *როგორიცაა დამოუკიდებლობა და თანაბარი განაწილება; გამოთვლების ეფექტურობა და მონაცემთა შენახვა; ნაკადების დიდი რაოდენობა და მათი დამოუკიდებლობა; კვლავ გენერირება-გამეორება.*

კვლავ-გენერირება და გამეორება (2.1)–(2.3) ტიპის გენერატორებში ადვილად მისაღწევია. ამისათვის საკმარისია z_0-ის დამახსოვრება და შემდგომში გენერატორის აქტივიზაცია დამახსოვრებული მნიშვნელობით, რათა მივიღოთ $\{\gamma_i\}_{i\ge 1}$ – იგივე მიმდევრობა. შეიძლება ციკლის გარკვეული ნაწილის გამეორებაც, თუ დამახსოვრებული იქნება შესაბამისი z_i მნიშვნელობა.

როგორც ყოველთვის, (2.1)–(2.3) ტიპის გენერატორებით ნაკადების გენერირება გარანტირებულია, თუ დამახსოვრებული იქნება მათი საწყისი მნიშვნელობები. მაგალითად, თუ ჩვენ გვჭირდება 100 000 რიცხვი სიგრძის მქონე ნაკადები, მაშინ z_0 პირველი ნაკადის

საწყისი მნიშვნელობაა, z_{100000} კი იქნება მეორე ნაკადის საწყისი მნიშვნელობა, z_{200000} – მესამე ნაკადის საწყისი მნიშვნელობა და ა.შ. ანალოგიურად ავაგებთ სხვადასხვა მოცულობის ნაკადებსაც.

ამ თავის შემდეგ პარაგრაფებში განიმარტება, თუ როგორ უნდა აირჩეს წკგ-ის პარამეტრები, რათა მიღებული იქნას "კარგი" გენერატორი. აგრეთვე განხილული იქნება უკიდურესად "ცუდი" გენერატორები, რომლებიც განსაკუთრებულ შემთხვევებში მაინც შეიძლება გამოყენებულ იქნას სიმულაციებსა და მოდელირებაში.

§2.2. წანაცვლებული გენერატორები

განიხილება შემთხვევა, როცა (2.1)–(2.3) მოდელში $c > 0$. თეორემა 2.1-ის ა) პირობის შესრულების შედეგად მიღწეული იქნება სრული პერიოდი. ჯერ შევარჩიოთ m მოდული. რომ მივიღოთ გრძელი პერიოდი და γ_i – გენერირებული რიცხვების მაღალი თანაბარი სიმკვრივე [0,1] ინტერვალში, აუცილებელია m მოდულმა მიიღოს დიდი მთელი მნიშვნელობა. გარდა ამისა (2.1)–(2.3) გენერატორში m-ზე გაყოფა და ნაშთის მიღება საკმაოდ გრძელი არითმეტიკული ოპერაციაა. ამიტომ სასურველია მოვერიდოთ ასეთი *გაყოფის პირდაპირ შესრულებას* [1],[2]. ცხადია, m-ის არჩევანი უნდა გავაკეთოთ 2-ის ხარისხებზე: $m = 2^b$ სახით, სადაც b კომპიუტერის სიტყვის ბიტების რაოდენობაა (სიტყვის, რომელშიც შესაძლებელია მონაცემთა შენახვა). ნაწილობრივ, ბევრ კომპიუტერსა და კომპილატორში გამოიყენება 32-ბიტიანი სიტყვა, ამასთან სასაზღვრო მარცხენა ბიტი გამოიყენება ნიშნისათვის. მაშასადამე $b = 31$ და $m = 2^{31}$. გარდა ამისა, თუ $b > 31$ და $m = 2^b$ (მაგ., $b = 2^{63}$), მაშინ ნამდვილად გვერდს ვუვლით m-ზე პირდაპირ გაყოფას უმეტეს რაოდენობა კომპიუტერებისთვის, რადგანაც 2^b-ზე გაყოფისას შესაძლებელია *მთელი რიცხვების გადავსების* გამოყენება. უდიდესი მთელი რიცხვი, რომელიც შეიძლება წარმოვადგინოთ, არის $2^b - 1$. ნებისმიერი მცდელობა უდიდესი მთელი N რიცხვის შენახვისა სიტყვაში, რომელსაც აქვს $h > b$ ორობითი ნიშნები, დასაშვებ b ნორმას სცილდება, დამთავრდება მარცხენა $h - b$ ორობითი ნიშნების დაკარგვით. სიტყვაში ჩაწერილი რიცხვი იქნება $N \pmod{2^b}$. შემდეგ მაგალითში ნაჩვენებია მოდულით გაყოფა, როდესაც გამოყენებულია სიტყვის გადავსება $m = 2^b$ წარმოდგენისთვის.

დავუშვათ, რომ წკგ წინა მაგალითისთვის სრულდება წარმოსახვით კომპიუტერზე, რომლის სიტყვაში ბიტების რიცხვია $b = 4$. შესაბამისად $m = 16 = 2^b$. როგორ გამოვიყენოთ *სიტყვის გადავსება*, რომ მივიღოთ $z_7 = 12$ $z_6 = 5$-სგან?

ამ შემთხვევაში $5z_6 + 3 = 28$ რიცხვის ორობითი წარმოდგენაა 11100. რადგან 4-ბიტიან კომპიუტერის სიტყვაში შესაძლებელია მხოლოდ 4 ორობითი ნიშნის შენახვა, მაშინ 11100-ს მარცხენა ერთი სასაზღვრო ციფრი 1 ჩამოცილდება და დარჩება ორობითი რიცხვი 1100, რომელიც წარმოადგენს $z_7 = 12$ ათობითი რიცხვის ორობით წარმოდგენას.

ჩვენ *გაფრთხილებთ მკითხველს*, რომ იმ კომპიუტერზე, რომელზეც შესრულდება გენერატორისთვის ჩაწერილი პროგრამა, აუცილებელია *შემოწმდეს მთელი რიცხვების გადავსების დამუშავების საშუალება*. მაგალითად, რიცხვის ნიშნის ბიტი შეიძლება ჩავრთოთ ოპერაციაში, რაც გამოიწვევს კორექტირებას, რომელიც შეიძლება დამოკიდებული იყოს კომპიუტერის არქიტექტურაზე, მთელი რიცხვის წარმოდგენის ფორმატზე და დაპროგრამების ენაზე.

ასე რომ $m = 2^b$ წარმოადგენს მოდულის კარგ არჩევანს. ასეთ შემთხვევაში თეორემა 2.1-ის გათვალისწინებით ჩვენ მივიღებთ სრულ პერიოდს, თუ c დადებითი კენტი რიცხვი იქნება და $a - 1$ გაიყოფა 4-ზე. შევნიშნოთ, რომ z_0 შეიძლება აირჩეს ნებისნიერად 0-დან $m - 1$-მდე და ეს არჩევანი გავლენას ვერ მოახდენს გენერატორის პერიოდზე.

ახლა განვიხილოთ a -ს არჩევის საკითხი. წკგ-ის კონსტრუირების ადრეულ ნაშრომებში აქცენტი კეთდებოდა a და z_{i-1} რიცხვების ნამრავლის ეფექტურობაზე, როდესაც a მამრავლი წარმოდგება ფორმით [2]:

$$a = 2^e + 1, \tag{2.6}$$

რაიმე e მთელი რიცხვისთვის. შესაბამისად

$$az_{i-1} = 2^e \cdot z_{i-1} + z_{i-1}.$$

ეს ნიშნავს, რომ az_{i-1} შეიძლება მივიღოთ z_{i-1} მთელი რიცხვის სიტყვაში ორობითი წარმოდგენის მარცხნივ e ბიტზე "წანაცვლებით" და შემდეგ მასზე z_{i-1}-ის მიმატებით. ე.ი. *პირდაპირი გადამრავლების ოპერაცია იცვლება წანაცვლებისა და შეკრების ოპერაციებით.* თუმცა მოგვიანებით გაირკვა (Knuth, 1998), რომ (2.6) ტიპის მამრავლების გამოყენება წკგ-ს ალგორითმებში მიუღებელია, რადგან გენერირებული შემთხვევითი რიცხვები გამოირჩევიან *ცუდი სტატისტიკური თვისებებით* (მეზობლების მაღალი კორელაცია და ა.შ.).

მაინც როგორ ავირჩიოთ a და c -ს წანაცვლებულ $(c > 0)$ წრფივ კონგრუენტულ გენერატორში, როცა $m = 2^b$? საკუთარი გამოცდილებიდან ვიცით, რომ ეს მარტივი ამოცანა არაა და საერთოდ შეიძლება ითქვას უარი წანაცვლებულ $(c > 0)$ გენერატორზე. უფრო მარტივი და გასაგებია მულტიპლიკატიური წკგ, რომელიც უკეთ მოქმედებს და უფრო ხშირად გამოიყენება.

§2.3. მულტიპლიკატიური გენერატორები

მულტიპლიკატიური წკგ-ის გამოყენება ხელსაყრელია, რადგან საჭირო არაა c პარამეტრის განსაზღვრა $(c = 0)$, მაგრამ მას არ გააჩნიათ სრული პერიოდი, რადგან თეორემა 2.1-ის ა) პირობა არ სრულდება (რამეთუ $m > 0$ მოდული და $c = 0$ იყოფიან m -ზე, ე.ი. ისინი ურთიერთმარტივები არაა). თუმცა შესაძლებელია პერიოდი $m-1$ აღმოჩნდეს (ამას მომავალში ვნახავთ), თუ სათანადოდ ავარჩევთ m და a -ს. მულტიპლიკატიურ წკგ-ს იყენებდნენ უფრო ადრე, ვიდრე წანაცვლებულ წკგ-ს. თანაც ამჟამად გამოყენებული უმრავლესობა წკგ-ებისა სწორედ რომ მულტიპლიკატიურია. რაც შეეხება წანაცვლებულ წკგ-ის *უპირატესობას, იგი დღემდე არაა დამტკიცებული* [1].

როგორც დამტკიცდა (Knuth, 1998), მულტიპლიკატიურ გენერატორებში, რომლებშიც მოდული $m = 2^b$, პერიოდი არ აღემატება 2^{b-2}-ს. ე.ი. მოდულის მეოთხედს (თუ z_0 კენტია და a -ს გააჩნია $8k+3$ ან $8k+5$ გამოსახულება $k = 0, 1, 2, \ldots$ რაიმე მთელი მნიშვნელობისთვის). ე.ი. $\{0, 1, \ldots, m-1\}$ მნიშვნელობებიდან მხოლოდ მეოთხედი რაოდენობა შეიძლება მივიღოთ z_i -ის მნიშვნელობებად, თანაც ცნობილი არაა ეს $\frac{m}{4}$ რაოდენობა რიცხვები როგორაა განაწილებული $(0,1)$-ზე. შეიძლება რომელიმე ორ მათგანს შორის დიდი მანძილიც იყოს, ამიტომ შემუშავდა სხვა მეთოდები m -ის ასაგებად.

$m = 2^b$ -ს ნაცვლად m -ის როლში იღებენ ისეთ უდიდეს მარტივ რიცხვს, რომელიც ნაკლებია 2^b -ზე [1, 2]. მაგალითად: თუ $b = 31$, მაშინ უდიდესი მარტივი მთელი რიცხვი, რომელიც ნაკლებია 2^b -ზე, შესაბამისად არის $m = 2^{31} - 1 = 2\,147\,783\,647$. პერიოდი იქნება $m-1$. ასეთ გენერატორებს ეწოდებათ მულტიპლიკატიური წკგ *მარტივი მოდულით.* a ისე უნდა აირჩეს, რომ $a^{m-1} - 1$ უნაშთოდ იყოფოდეს m -ზე (აქ $m = 2^{31} - 1$)! მულტიპლიკატიური გენერატორის გამოყენებასთან წარმოიშობა პრობლემა: სიტყვაში *გადავსების მექანიზმი ვერ გამოიყენება პირდაპირი გადამრავლების ოპერაციის ნაცვლად* $\bmod m$ -ის გაყოფისას, რადგან აქ უკვე აღარ ვირჩევთ $m = 2^b$!

ამ პრობლემის დასაძლევად შემოთავაზებულია (Morse and Roberts, 1983, და სხვ.) მეთოდები, რომლებიც იყენებენ ე.წ. "*სიმულაციურ გაყოფას*", რომელიც გვერდს უვლის პირდაპირ გაყოფას. ასეთ გენერატორებში მარტივი m მოდულის როლში იყენებენ

$$2^b - q \tag{2.7}$$

ტიპის გამოსახულებებს ($q > 0$ მთელი რიცხვია), რაც საშუალებას გვაძლევს *გაყოფა დავიყვანოთ გადავსების მექანიზმზე*: რომ მივიღოთ $z_i = (az_{i-1}) (\mathrm{mod}\ 2^b - q)$ გარდაქმნა z_{i-1}-დან, დავუშვათ, რომ $z_i' = (az_{i-1}) (\mathrm{mod}\ 2^b)$ შეიძლება გამოითვალოს გადავსების მექანიზმით პირდაპირი გაყოფის გარეშე. თუ k უდიდესი მთელი რიცხვია, რომელიც ნაკლებია ან ტოლი $az_{i-1}/2^b$ გამოსახულებაზე (ეს უკანასკნელი კი მარჯვნივ ძვრის ოპერაციით შეიცვლება), მაშინ

$$z_i = \begin{cases} z_i' + kq, & \text{თუ} \quad z_i' + kq < 2^b - q, \\ z_i' + kq - (2^b - q), & \text{თუ} \quad z_i' + kq \geq 2^b - q. \end{cases} \tag{2.8}$$

მარსესა და რობერტსის გენერატორში (რომლის C-პროგრამულ კოდს გთავაზობთ §2.6-ში) გამოყენებულია სიმულაციური გაყოფა. პარაგრაფის ბოლოში გთავაზობთ პარამეტრების სარეკომენდაციო მნიშვნელობებს მულტიპლიკატიურ წკგ-ში:

1. $m = 2^{31} - 1, \quad a = 16\,807$ ან $a = 630\,360\,016$.
2. ორობითი თანრიგიანი კომპიუტერისათვის $m = 2^b$, $b \geq 4$, z_0 და m ურთიერთმარტივია და $a = 3 \pmod 8$ ან $a = 5 \pmod 8$, მაშინ დასაშვებ a პარამეტრთა მეოთხედი იძლევა სრულ პერიოდს.
3. ათობითი თანრიგიანი კომპიუტერისათვის $m = 10^b$, $b \geq 5$, z_0 არაა ჯერადი 2-ის ან 5-ის. მაშინ პერიოდია $5 \cdot 10^{b-2}$, თუ $a \pmod{200}$ ღებულობს ერთ-ერთ შემდეგ მნიშვნელობას: 3, 11, 13, 19, 21, 27, 29, 37, 53, 59, 61, 67, 69, 83, 91, 109, 117, 123, 131, 139, 141, 147, 163, 171, 173, 179, 181, 187, 189, 197.

§2.4. სხვა ტიპის გენერატორები

წინა პარაგრაფებში განხილული იყო პრაქტიკაში ფართოდ გამოყენებული წკგ. თუმცა მათ ალტერნატიულად არსებობს მრავალი სხვა ტიპის გენერატორი. განვიხილავთ მხოლოდ ძირითად ალტერნატიულ გენერატორებს.

§2.4.1. ზოგადი ტიპის კონგრუენტული გენერატორები

ზოგადი ტიპის კონგრუენტული გენერატორების განმსაზღვრელი ფორმულაა

$$z_i = g(z_{i-1}, z_{i-2}, \ldots) \pmod m, \tag{2.9}$$

სადაც g წარმოადგებს z_{i-1}, z_{i-2} და ა.შ. წინა გენერირებული სიდიდეების დეტერმინისტულ ფუნქციას. ცხადია, რომ (2.9) ფორმულით განსაზღვრული z_i რიცხვები $\{0, 1, \ldots, m-1\}$ სიმრავლიდანაა და რომ მივიღოთ $U(0,1)$ კანონით განაწილებული შერჩევა, საჭიროა მათი m-ით ნორმირება:

$$\gamma_i = z_i / m. \tag{2.10}$$

(ცხადია, რომ წრფივი კონგრუენტული გენერატორი განეკუთვნება ზოგადი ტიპის კონგრუენტულ გენერატორებს, რადგან g წრფივი ფუნქციაა: $g(z_{i-1}) = az_{i-1} + c$). განვიხილავთ ზოგიერთ ასეთი ტიპის გენერატორებს.

2.4.1.1. *კვადრატული კონგრუენტული გენერატორები* მოიცემა ფორმულით

$$g(z_{i-1}) = a' z_{i-1}^2 + az_{i-1} + c\,. \tag{2.11}$$

განსაკუთრებულ ყურადღებას იმსახურებს შემთხვევა: $a' = a = 1,\ c = 0$ და $m = 2^b$. მაქსიმალური პერიოდი აქაც m-ია.

2.4.1.2. *მრავალჯერადი რეკურსიული გენერატორი* განისაზღვრება ფორმულით:

$$g(z_{i-1}, z_{i-2}, \ldots) = a_1 z_{i-1} + a_2 z_{i-2} + \cdots + a_q z_{i-q}\,, \tag{2.12}$$

სადაც $a_1, a_2, \ldots, a_q$ მუდმივებია, q ნატურალური რიცხვია. არსებობს შესაძლებლობა მივიღოთ უდიდესი პერიოდი $m^q - 1$, თუ შესაბამისად ავირჩევთ პარამეტრებს. ასეთი ტიპის გენერატორები შესწავლილია ლეკიუერის და ბლუინის (L'Ecuyer and Blouin, 1988) მიერ. (2.12) ტიპის გენერატორებს მიეკუთვნებათ "ძველი" – ფიბონაჩის გენერატორი:

$$z_i = (z_{i-1} + z_{i-2}) \pmod m\,. \tag{2.13}$$

m-ის სათანადოდ შერჩევით ფიბონაჩის გენერატორის პერიოდი შეიძლება აღემატებოდეს m-ს, თუმცა მტკიცდება, რომ სტატისტიკური თვალსაზრისით იგი *სრულიად გამოუყენებელია.*

§2.4.2. რთული გენერატორები

მკვლევარებმა შეიმუშავეს მეთოდები, რომლებიც იყენებენ ერთ ან რამოდენიმე შემთხვევით რიცხვთა გენერატორს. ისინი კომბინირებენ საბოლოო შემთხვევითი რიცხვების მისაღებად. ასეთ გენერატორებს *რთული გენერატორები* უწოდეს.

2.4.2.1. *ლეკიუერის გენერატორი.* ლეკიუერის (L'Ecuyer, 1988) მიერ შემუშავდა კომბინირებული გენერატორი, რომელიც ორი მარტივი გენერატორის გაერთიანებას გულისხმობს: ვთქვათ ამ გენერატორებით მიღებული მთელი რიცხვების მიმდევრობებია $\{z_{1i}\}$ და $\{z_{2i}\}$. დავუშვათ, რომ

$$z_i = (z_{1i} - z_{2i}) \pmod m \tag{2.14}$$

რომელიმე მთელი m-სთვის. ამ გენერატორს აქვს ერთი დიდი უპირატესობა: მას შეიძლება აღმოაჩნდეს საკმაოდ დიდი პერიოდი (ავტორის მაგალითში ის 10^{18}-ს შეადგენს), თანაც შემადგენელი ორი გენერატორის მამრავლებში შეიძლება "პატარა" რიცხვები იყოს. როგორც ცდებმა აჩვენა, გენერატორი მოქმედებს საკმაოდ სწრაფად და გააჩნია კარგი სტატისტიკური თვისებები.

ამ მეთოდის განზოგადოება კვლავ ლეკიუერს (1999) ეკუთვნის: თუ გვაქვს $\{z_{1i}\}, \{z_{2i}\}, \ldots, \{z_{ji}\}$ რაოდენობა გენერატორი, მაშინ საბოლოო შემთხვევითი მთელი რიცხვი გამოითვლება ფორმულით:

$$\begin{cases} z_i = (\delta_1 z_{1i} + \delta_2 z_{2i} + \cdots + \delta_j z_{ji}) \pmod{m_1}\,, \\ \gamma_i = z_i / m_1\,. \end{cases} \tag{2.15}$$

ლეკიუერმა ჩაატარა ფართო მასშტაბული კვლევა (2.15) ტიპის გენერატორებისა და გამოავლინა პარამეტრთა ის მნიშვნელობები, რომლებიც იძლევიან დიდ პერიოდს და სტატისტიკურად ვარგისია. მან ასევე წარმოადგინა C-პროგრამა ერთ-ერთი მუშა გენერატორისთვის [1]. ერთ-ერთი ასეთი მარტივი მოდელია შემდეგი გენერატორი:

$$\begin{cases} z_{1,i} = (1403\,580\, z_{1,i-2} - 810\,728\, z_{1,i-3}) \pmod{(2^{32} - 209)} \\ z_{2,i} = (527612\, z_{2,i-1} - 1370589\, z_{2,i-3}) \pmod{(2^{32} - 22853)} \\ z_i = (z_{1,i} - z_{2,i}) \pmod{(2^{32} - 209)}\,, \\ \gamma_i = z_i / (2^{32} - 209)\,. \end{cases} \tag{2.16}$$

წარმოდგენილ გენერატორს გააჩნია დაახლოებით 2^{191}-ის ტოლი პერიოდი (დაახლოებით $3{,}1\cdot 10^{57}$) და შესანიშნავი სტატისტიკური თვისებები. მისი საწყისი მნიშვნელობებია:

$$z_{10},\ z_{11},\ z_{12},\ z_{20},\ z_{21},\ z_{22}\ . \tag{2.17}$$

(2.16)-ით განსაზღვრული პირველი შემთხვევითი რიცხვია γ_3! (წარმოდგენილი (2.16) ალგორითმის კოდი C-ენაზე იხ. [1]-ში.)

2.4.2.2. *ტაუსვორტის გენერატორი.* რამოდენიმე საინტერესო გენერატორი იქნა აგებული ტაუსვორტის (Tausworthe, 1965) შრომების საფუძველზე. ეს გენერატორები, რომლებიც დაკავშირებულია *კრიპტოგრაფიულ მეთოდებთან*, უშუალოდ მოქმედებენ ბიტებზე შემთხვევითი რიცხვების მისაღებად.

ავაგოთ $b_1, b_2, \ldots$ – ბინარული ნიშნების რეკურენტული მიმდევრობა

$$b_i = (c_1 b_{i-1} + c_2 b_{i-2} + \cdots + c_q b_{i-q}) \pmod 2 \tag{2.18}$$

ფორმულით, სადაც $c_1, c_2, \ldots, c_q$ მუდმივებია 1 ან 0 მნიშვნელობებით. ტაუსვორტის (2.18) ალგორითმის გენერატორები პრაქტიკაში ძირითადად შემდეგი სახით გამოიყენება:

$$b_i = (b_{i-r} + b_{i-q}) \pmod 2\,, \tag{2.19}$$

სადაც r და q მთელი რიცხვებია $0 < r < q$ პირობით. (2.19) ფორმულაში შეკრებისა და $\bmod 2$ -ის აღების ოპერაცია საკმაოდ სწრაფია, რადგან ის ექვივალენტურია ბიტებზე "ან ოპერაციის გამორიცხვის" ოპერაციისა:

$$b_i = \begin{cases} 0\,, & \text{თუ} \quad b_{i-r} = b_{i-q}\,, \\ 1\,, & \text{თუ} \quad b_{i-r} \neq b_{i-q}\,. \end{cases} \tag{2.20}$$

რომ ამუშავდეს (2.19) ალგორითმი, აუცილებელია პირველი q რაოდენობა $b_1, b_2, \ldots, b_q$ ბიტების ინიციალიზაცია (განსაზღვრა). მოვიყვანოთ მაგალითი (Lewis and Payne, 1973).

მაგალითი. ვთქვათ (2.19) მოდელში $r = 3$ და $q = 5$, ხოლო საწყისი მნიშვნელობებია $b_1 = b_2 = \cdots = b_5 = 1$. თუ $i \geq 6$, მაშინ

$$b_i = \begin{cases} 0\,, & \text{თუ} \quad b_{i-3} = b_{i-5}\,, \\ 1\,, & \text{თუ} \quad b_{i-3} \neq b_{i-5}\,. \end{cases}$$

პირველი 42 ორობითი ნიშანი $\{b_i\}_i$ მიმდევრობისა ასეთია

$$\overbrace{11111000110111010100001001011001}^{\text{პერიოდის სიგრძე 31 ბიტია}}11111000110\,.$$

შევნიშნოთ, რომ პერიოდი ბიტებში შეადგენს $2^q - 1 = 31$-ს.

ტაუსვორტის ტიპის ალგორითმებში ერთადერთი პრობლემაა: როგორ გარდაიქმნან ისინი შემთხვევით რიცხვებში $U(0,1)$ განაწილებული კანონით? ერთი ყველაზე ბუნებრივი გზაა b_i ნიშნების გაერთიანება რაიმე l სიგრძის ბიტურ წარმოდგენებში, რათა მივიღოთ მთელი რიცხვები 0-დან $2^l - 1$ -მდე, რომელთა 2^l -ზე გაყოფით მივიღებთ γ – თანაბრად განაწილებულ შემთხვევით რიცხვებს. მოცემულ მაგალითში თუ $l = 4$, მაშინ მივიღებთ შემდეგ შემთხვევით რიცხვებს:

$$\frac{15}{16},\ \frac{8}{16},\ \frac{13}{16},\ \frac{13}{16},\ \frac{4}{16},\ \frac{2}{16},\ \frac{5}{16},\ \frac{9}{16},\ \frac{15}{16},\ \frac{1}{16}, \ldots.$$

შევნიშნოთ, რომ l -მა არ უნდა გადააჭარბოს კომპიუტერის სიტყვის ზომას.

r და q პარამეტრების შერჩევის საკითხი შესწავლილია ტაუსვორტისა და სხვათა მიერ [1].

ტაუსვორტის გენერატორებს გააჩნიათ მთელი რიგი უპირატესობანი წრფივ კონგრუენტულ გენერატორებთან მიმართებაში. ტაუსვორტის გენერატორებს გააჩნიათ გაცილებით

მეტი სიგრძის პერიოდები (მაგ.: $2^{521}-1>10^{156}$ და მეტი). მათ ასევე გააჩნიათ თეორიული ტესტირები კარგი საფუძვლები. ცნობილი მკვლევარი ჯონსონი (Johnson, 1987) რეკომენდაციას აძლევს ტაუსვორტის ტიპის გენერატორების გამოყენებას, ვიდრე წკგ-ისა მრავალგანზომილებიანი მოდელირების ამოცანებში. მაგრამ სტატისტიკური ტესტების მიმართ ისინი ნაკლებად მდგრადია და ამიტომაც პრაქტიკაში ნაკლები გამოყენება ჰპოვეს, თუმცა კი წარმატებით გამოიყენებიან კრიპტოგრაფიის ამოცანებში.

2.4.2.3. *ბლუმებისა და შუბას* (Blum, Blum and Shub, 1986) *გენერატორი* წარმოადგენს კიდევ ერთ რთულ გენერატორს, რომელიც ფართოდ გამოიყენება კრიპტოგტაფიაში: ვთქვათ p და q დიდი მარტივი რიცხვებია, რომელთათვისაც $p-3$ და $q-3$ იყოფიან 4-ზე. განისაზღვრება $m=pq$ მოდული და კვადრატული კონგრუენტული გენერატორი

$$z_i = z_{i-1}^2 \pmod m, \tag{2.21}$$

$\{z_i\}$ გენერირებული მიმდევრობიდან განისაზღვრება $\{b_i\}_i$ ბიტების მიმდევრობა ლუწობაზე კონტროლის მეთოდით: $b_i=1$, თუ z_i ლუწია და $b_i=1$, თუ z_i კენტია.

ავტორების მიერ ნაჩვენებია რომ ასეთი ალგორითმით მიღებული $b_1, b_2, \ldots, b_n, \ldots$ ბიტების მიმდევრობის დადგენა პრაქტიკულად შეუძლებელია, რადგან მასში არაშემთხვევითობის აღმოჩება ძალიან ძნელია, რაც ექვივალენტურია m რიცხვის pq ნამრავლად წარმოდგენაში, რაც კომპიუტერისაგან ითხოვს ძალიან დიდ რესურსებს. ასეთი თეორიული უზრუნველყოფა კრიპტოგრაფიის კოდების დაშიფრვაში რა თქმა უნდა ძალიან მნიშვნელოვანია.

§2.5. შემთხვევითი რიცხვების გენერატორების ტესტირება

როგორც წინა პარაგრაფებში გაირკვა ყველა შემთხვევით რიცხვთა გენერატორები, რომელიც კომპიუტერულ მოდელირებაში გამოიყენება, *სრულიად დეტერმინისტულია.* ჩვენ მხოლოდ ვიმედოვნებთ, რომ გენერირებული $\{\gamma_i\}_{i\geq 1}$ მნიშვნელობები გამოიჩენენ ისეთ თვისებებს, თითქოს ისინი იყვნენ დამოუკიდებელნი და თანაბრად განაწილებულნი $U(0,1)$ კანონით.

ამ პარაგრაფში ჩვენ განვიხილავთ რამოდენიმე ტესტს, რომელიც შეიძლება გამოყენებულ იქნას გენერირებული $\{\gamma_i\}_{i\geq 1}$ მნიშვნელობების შესამოწმებლად დამოუკიდებლობაზე და თანაბრად განაწილებაზე.

კომპიუტერთა უმრავლესობის პროგრამულ უზრუნველყოფაში შედის *"ჩადგმული" შემთხვევით რიცხვთა გენერატორი. დაჟინებით რეკომენდაციას ვუწევთ,* რომ კომპიუტერულ მოდელირებაში მათი გამოყენების წინ შეფასდეს მათი მოქმედების სიზუსტე და თუ რა გენერატორია ის (ჩვენთვის უკვე ცნობილი კლასიფიკაციით), როგორია მისი რიცხვითი პარამეტრები. თუ "ჩადგმული" გენერატორი არ აღმოჩნდება ერთ-ერთი კარგი გენერატორი, რომელიც ჩვენ უკვე განვიხილეთ, ან კიდევ იგი უცნობი აღმოჩნდება ჩვენთვის, მაშინ აუცილებელია მისი ტესტირება ემპირიული ტესტებით.

§2.5.1. ემპირიული ტესტები

გენერატორების *ემპირიული ტესტირების მიზანია* განისაზღვროს გენერირებული $\{\gamma_i\}_{i\geq 1}$ რიცხვები თუ რამდენად თანხვდენაშია და მსგავსია დამოუკიდებელი და $(0,1)$-ზე თანაბრად განაწილებული შემთხვევით სიდიდეებთან.

აქ განვიხილავთ 4 *ასეთ მთავარ ტესტს* (დანარჩენი ტესტების ნახვა შეიძლება კნუტის (Knuth, 1998) ნაშრომში.

I ტესტი: შემოწმება თანაბრად განაწილებაზე

როგორც სტატისტიკის კურსიდანაა ცნობილი, რაიმე $\gamma_1, \gamma_2, \ldots, \gamma_n$ შერჩევის განაწილების დადგენა ხერხდება "ხი-კვადრატ" კრიტერიუმით:

[0,1] ინტერვალს ვყოფთ k რაოდენობა თანაუკვეთი ტოლი სიგრძის ქვეინტერვალთა გაერთიანებად: გენერატორით ვახდენთ $\gamma_1, \gamma_2, \ldots, \gamma_n$ რიცხვების გენერირებას (n მეტი ან ტოლი უნდა იყოს 100-ზე, ხოლო n/k ტოლი უნდა იყოს სულ მცირე 5-ის). დავუშვათ, რომ f_j $(j = 1, 2, \ldots, k)$ არის $\{\gamma_i\}_{i\geq 1}$ რიცხვების აბსოლუტური სიხშირეები j -ურ ქვეინტერვალში მოხვედრისას. შევადგინოთ შემდეგი გამოსახულება:

$$\chi^2 = \frac{k}{n}\sum_{j=1}^{k}\left(f_i - \frac{n}{k}\right)^2 \sim \chi^2(k-1). \qquad (2.22)$$

ცნობილია, რომ დიდი n -სთვის $(n \geq 100)$ χ^2 -ს აქვს დაახლოებით "ხი-კვადრატ" განაწილება $(k-1)$ თავისუფლების ხარისხით.

ჩამოვაყალიბოთ H_0 – ნულოვანი ჰიპოთეზა: $\{\gamma_i\}_{i\geq 1}$ წარმოადგენს დამოუკიდებელ და $U(0,1)$ კანონით განაწილებულ შერჩევას რაიმე α მნიშვნელოვნების დონით (I გვარის შეცდომის ალბათობით). მაშინ ჩვენ ვუარყოფთ ამ ჰიპოთეზას, თუ

$$\chi^2 > \chi^2_{k-1,\alpha}, \qquad (2.23)$$

სადაც $\chi^2_{k-1,\alpha}$ წარმოადგენს $(k-1)$ თავისუფლების მქონე χ^2 განაწილების α დონის ზედა კრიტიკულ წერტილს. თუ (2.23) უტოლობა არ სრულდება, მაშინ არ არსებობს არავითარი ობიექტური წინაპირობა H_0 ჰიპოთეზის უარყოფისა. k -ს დიდი მნიშვნელობისთვის დასაშვებია მიახლოება:

$$\chi^2_{k-1,\alpha} \approx (k-1)\left\{ 1 - \frac{2}{9(k-1)} + z_\alpha \cdot \sqrt{\frac{2}{9(k-1)}} \right\}^3,$$

სადაც z_α ნორმალური განაწილების α დონის ზედა კრიტიკული წერტილია.

მაგალითი. ჩვენ განვიხილავთ

$$z_i = 630\,360\,016 \cdot z_{i-1} \pmod{2^{31}-1} \qquad (2.24)$$

წკგ-ის შემოწმებას თანაბარ განაწილებაზე (გამოიყენეთ §2.6-ის C-პროგრამა და აიღეთ ნაკადი 1). ამისთვის ავიღოთ $k = 2^{12} = 4096$ (თანაბარ განაწილებაზე შევამოწმოთ ყველაზე მნიშვნელოვანი პირველი თორმეტი ბიტი γ_i მნიშვნელობებისა). n -ის როლში ავიღოთ: $n = 2^{15} = 32768$. მაშინ გამოთვლებით მივიღებთ (შეამოწმეთ!), რომ $\chi^2 = 4141{,}0$. $\alpha = 0{,}1$ მნიშვნელოვნების დონით

$$\chi^2_{4095;0.1} = 4211{,}4.$$

შესაბამისად $\alpha = 0{,}1$ მნიშვნელოვნების დონით H_0 ჰიპოთეზა მიღებულია $(\chi^2 < \chi^2_{4095;0.1})$. ანუ 2^{15} რეალიზაცია $\{\gamma_i\}$ მნიშვნელობებისა შეიძლება ჩაითვალოს თანაბარი განაწილების შერჩევად.

II ტესტი: შემოწმება სერიების კრიტერიუმით

ფაქტიურად ეს ტესტი წარმოადგენს χ^2 -კრიტერიუმის განზოგადოებას მაღალ განზომილებებში. კერძოდ, თუ $\{\gamma_i\}_{i=1}^n$ გენერირებული მნიშვნელობანი მართლა წარმოადგენს $U(0,1)$ კანონით განაწილებულ შერჩევას, მაშინ რაიმე d -განზომილებიანი კორტეჟები

$$\gamma_1' = (\gamma_1, \gamma_2 \ldots, \gamma_d), \qquad \gamma_2' = (\gamma_{d+1}, \gamma_{d+2} \ldots, \gamma_{2d}), \ \ldots \qquad (2.25)$$

უნდა იყოს d -განზომილებიან $[0,1]^d$ ერთეულოვან ჰიპერკუბში თანაბრად განაწილებული შემთხვევითი ვექტორის შერჩევა. ამის შესამოწმებლად შევადგინოთ ტესტი:

ავიღოთ d რაოდენობა $[0,1]$ ინტერვალი და თითოეული დავყოთ k რაოდენობა ტოლი სიგრძის მქონე თანაუკვეთი ინტერვალების გაერთიანებად, მოვახდინოთ

$$\gamma_1',\ \gamma_2',\dots,\gamma_n' \tag{2.26}$$

ვექტორის შერჩევის გენერირება (სერიებად დაწყობა). ამისთვის საჭიროა nd რაოდენობის გენერირების შესრულება. ვთქვათ $f_{j_1,j_2,\dots,j_d}$ არის იმ ვექტორების რიცხვი (2.25)-დან, რომლის პირველი კომპონენტი ეკუთვნის პირველი ინტერვალის j_1 -ურ ქვეინტერვალს, მეორე კომპონენტი ეკუთვნის მეორე ინტერვალის j_2 -ურ ქვეინტერვალს და ა.შ. ვთქვათ n დიდია და დავუშვათ, რომ

$$\chi^2(d)=\frac{k^d}{n}\sum_{j_1=1}^{k}\sum_{j_2=1}^{k}\cdots\sum_{j_d=1}^{k}\left(f_{j_1,j_2,\dots,j_d}-\frac{n}{k^d}\right)^2 \tag{2.27}$$

განაწილებულია დაახლოებით χ^2 -კანონით, რომლის თავისუფლების ხარისხია k^d-1 (სასურველია, რომ $\frac{n}{k^d}\geq 5$).

d -განზომილებიან თანაბარ განაწილებაზე შემოწმება შესრულდება ანალოგიურად I ტესტისა.

მაგალითი. შევამოწმოთ ნულოვანი ჰიპოთეზა იმაზე, რომ გენერირებულ შერჩევაში

$$(\gamma_1,\gamma_2),\ (\gamma_3,\gamma_4),\ \dots,\ (\gamma_{2n-1},\gamma_{2n})$$

წყვილები დამოუკიდებლებია და ერთნაირად განაწილებული ვექტორებია $d=2$ განზომილების ერთეულოვან კვადრატში თანაბარი განაწილებით.

ამისთვის გამოვიყენოთ წინა მაგალითის გენერატორი, ოღონდ ნაკადი 2-დან. ავიღოთ $n=32\,768$ გენერირებული რიცხვი. ვთქვათ $k=64$. აქაც გვაქვს $64^2-1=4095$ თავისუფლების ხარისხი. $\alpha=0.1$ მნიშვნელოვნების დონის კრიტიკული წერტილია $\chi^2_{4095;0.1}=4211{,}4$. გამოთვლები გვიჩვენებს (C-ზე დაწერეთ კოდი და შეამოწმეთ!), რომ $\chi^2(2)=4016{,}5$, რაც ადასტურებს ნულოვან ჰიპოთეზას; წყვილური შერჩევა თანაბარია. ანალოგიურად შეიძლება განიხილოთ $d=3,4$ და ა.შ. შემთხვევები (დაწერეთ კოდი C-ზე და განიხილეთ!).

მაღალ განზომილებებში შემოწმება თანაბარ განაწილებაზე ჩვენ გვაინტერესებს შემდეგი მოსაზრებებიდან: თუ γ_i გენერირებული სიდიდეები დაკორელირებულია, მაშინ γ_i' – d -განზომილებიანი შემთხვევითი ვექტორის რეალიზაციები გადახრილი იქნება d -განზომილებიანი თანაბარი განაწილებიდან. თანაც d -განზომილებიან თანაბარ განაწილებაზე ჰიპოთეზის შემოწმება საშუალებას იძლევა ირიბად შევამოწმოთ ცალკეული γ_i გენერირებული სიდიდეების თანაბარი განაწილებაც. მაგალითად, თუ $\{\gamma_i,\gamma_{i+1}\}$ წყვილების შემოწმებაში შეინიშნება დადებითი კორელაცია, მაშინ ისინი დამოკიდებულებიც იქნებიან, რაც გენერატორის სტატისტიკური ტესტირების უარყოფით შედეგს ადასტურებს. შევნიშნოთ, რომ როცა $d>3$, მაშინ ტესტირება მოითხოვს საკმაოდ დიდ მეხსიერებას იმისთვის, რომ გამოვთვალოთ k^d რაოდენობა $f_{j_1,j_2,\dots,j_d}$ აბსოლუტური სიხშირეები.

III ტესტი: სერიების კრიტერიუმით შემოწმება დამოუკიდებლობაზე

ამ ტესტით ძირითადში მოწმდება გენერირებული $\{\gamma_i\}_{i=1}^n$ სიდიდეების მხოლოდ დამოუკიდებლობა (ე.ი. ვამოწმებთ მხოლოდ დამოუკიდებლობაზე და არა თანაბარ განაწილებაზე). $\{\gamma_i\}_{i=1}^n$ მიმდევრობა დავყოთ ე.წ. *ზრდადი სერიების გაერთიანებად*, თანაც ყოველ სერიაში γ_i

მნიშვნელობები მკაცრად იზრდებიან. ასე წარმოდგენილ შერჩევას *ზრდადი სერიების შერჩევა* ეწოდება. მაგალითად: ვთქვათ $\{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ სიდიდეების მაგალითია

$$0.86,\quad 0.11,\quad 0.23,\quad 0.03,\quad 0.13,\quad 0.06,\quad 0.55,\quad 0.64,\quad 0.87,\quad 0.10\ .$$

ეს შერჩევა იწყება 1-ის ტოლი სიგრძის ზრდადი სერიით (0.86). მას მიყვება 2-ის ტოლი სიგრძის ზრდადი სერია $(0.11,\ 0.23)$, შემდეგ კიდევ ერთი სერია 2-ის ტოლი სიგრძით $(0.03,\ 0.13)$, შემდეგ სერია 4-ის ტოლი სიგრძით $(0.06,\ 0.55,\ 0.64,\ 0.8)$ და ბოლოს 1-ის ტოლი სიგრძის სერია (0.10).

განვსაზღვროთ სიხშირეები 1, 2, 3, 4, 5 და ≥ 6 სიგრძის სერიებისა:

$$r_i = \begin{cases} i\text{-ს ტოლი სიგრძის სერიების რიცხვი}, & \text{თუ } i = 1, 2, 3, 4, 5, \\ 6\text{-ზე მეტი ან ტოლი სიგრძის სერიების რიცხვი}, & \text{თუ } i = 6 \end{cases}$$

(აქ წარმოდგენილი შერჩევისთვის გვაქვს $r_1 = 2,\ r_2 = 2,\ r_3 = 0,\ r_4 = 1,\ r_5 = 0,\ r_6 = 0$). სტატისტიკური კრიტერიუმი ასეთია (Knuth, 1998) [2]:

$$R = \frac{1}{n} \sum_{i=1}^{6} \sum_{j=1}^{6} a_{ij} (r_i - nb_i)(r_j - nb_j)\,, \tag{2.28}$$

სადაც a_{ij} ელემენტების მატრიცა ასეთია:

$$\begin{bmatrix} 4\,529.4 & 9\,044.9 & 13\,568 & 18\,091 & 22\,617 & 27\,892 \\ 9\,044.0 & 18\,097 & 27\,139 & 36\,187 & 45\,234 & 55\,789 \\ 13\,568 & 27\,139 & 40\,721 & 54\,281 & 67\,852 & 83\,685 \\ 18\,091 & 38\,187 & 54\,281 & 72\,414 & 90\,470 & 111\,580 \\ 22\,615 & 45\,234 & 67\,852 & 90\,470 & 113\,262 & 139\,476 \\ 27\,892 & 55\,789 & 83\,685 & 111\,580 & 139\,476 & 172\,860 \end{bmatrix},$$

ხოლო b_i სიდიდეები განისაზღვრება:

$$(b_1, b_2, b_3, b_4, b_5, b_6) = \left(\frac{1}{6}, \frac{5}{24}, \frac{11}{120}, \frac{19}{720}, \frac{29}{5040}, \frac{1}{840} \right).$$

როგორც კნუტის (Knuth) რეკომენდაციაა, $n \geq 4000$. ასეთ შემთხვევაში R სტატისტიკური კრიტერიუმი დაახლოებით განაწილებული იქნება χ^2 -კანონით 6-ის ტოლი თავისუფლების ხარისხით. H_0 ნულოვანი ჰიპოთეზა გულისხმობს, რომ $\gamma_1, \gamma_2, \ldots, \gamma_n$ მნიშვნელობები დამოუკიდებელია და ერთნაირად განაწილებული.

მაგალითი. განვიხილოთ §2.6-ის C კოდიდან მიღებული შერჩევა, ნაკადი 4, $n = 5000$. გამოთვლებით ვღებულობთ (შექმენით კოდი C-ზე და შეამოწმეთ):

$$(r_1, r_2, r_3, r_4, r_5, r_6) = (808, 1026, 448, 139, 43, 4),$$

ხოლო კრიტერიუმის მნიშვნელობაა $R = 9.3$. რადგან $\chi^2_{6;0.1} = 10$ -ს (10%-იანი მნიშვნელოვნების დონე) და $R < \chi^2_{6;0.1}$, ამიტომ ვთვლით, რომ 10%-იანი მნიშვნელოვნების დონით 5000 – გენერირებული მიმდევრობა დამოუკიდებელია.

შევნიშნოთ, რომ *ზრდადი სერიების კრიტერიუმი* შეიძლება შეიცვალოს *კლებადი სერიების* კრიტერიუმებით, თანაც a_{ij} და b_i კოეფიციენტები უცვლელი დარჩება. როგორც კნუტი თვლის, მონოტონური სერიების კრიტერიუმი მძლავრი კრიტერიუმია, რადგან მრავალი გენერატორი წარმატებით გადის I – χ^2 -კრიტერიუმს, მაგრამ მოდელირებაში მაინც დამაკმაყოფილებელ გამოყენებას ვერ პოულობს. ამის ერთ-ერთი მიზეზი კი ისაა, რომ უმრავლესობა ასეთი გენერატორებისა ვერ გადის III ტესტს.

IV ტესტი: კორელაციის ხარისხის შეფასების ტესტი

ამ ტესტის ამოცანაა შეაფასოს, გააჩნია თუ არა მაღალი კორელაცია გენერირებულ $\{\gamma_i\}_{i=1}^n$ შერჩევას.

ამისთვის ვიქცევით ასე: რაიმე l ნატურალური რიცხვისთვის ვითვლით $j=1,2,\ldots,l$ ინტერვალებით შერჩევით კორელაციებს. როგორც ცნობილია, j ინტერვალის კორელაცია გამოითვლება ფორმულით

$$\rho_j = c_j / c_0 , \tag{2.29}$$

სადაც

$$c_j = \operatorname{cov}(\gamma_i, \gamma_{i+j}) = \mathsf{E}(\gamma_i\, \gamma_{i+j}) - \mathsf{E}(\gamma_i)\mathsf{E}(\gamma_{i+j})$$

კოეფიციენტები (i -ინდექსზე არაა დამოკიდებული, რადგან ჩვენ ვუშვრბთ, რომ გენერირების პროცესი გვაძლევს ერთნაირად განაწილებულ რეალიზაციებს) აფასებენ მიმდევრობით j სიგრძის ინტერვალის კოვარიაციას $(c_0 = D(\gamma_i))$. თუ დავუშვებთ, რომ γ_i -ს თანაბარი განაწილება აქვს $(0,1)$-ზე ტესტი III-ის გავლის შემდეგ, მაშინ $\mathsf{E}(\gamma_i) = \mathsf{E}(\gamma_{i+j}) = \frac{1}{2}$, $\mathrm{D}(\gamma_i) = c_0 = \frac{1}{12}$, მაშასადამე კოვარიაცია გამოითვლება $c_j = \mathrm{E}(\gamma_i\, \gamma_{i+j}) - \frac{1}{4}$ ფორმულით და შესაბამისად კორელაციის კოეფიციენტი კი

$$\rho_j = 12 \cdot \mathrm{E}(\gamma_i\, \gamma_{i+j}) - 3 \tag{2.30}$$

ფორმულით. ρ_j -ს შერჩევითი შეფასებისთვის დაგვჭირდება $\gamma_1, \gamma_{1+j}, \ldots$ გენერირებული სიდიდეები

$$\hat{\rho}_j = \frac{12}{h+1} \cdot \sum_{k=0}^{h} \gamma_{1+kj}\, \gamma_{1+(k+1)j} - 3 , \tag{2.31}$$

სადაც $h = \left\lfloor \frac{n-1}{j} \right\rfloor - 1$ ($\lfloor t \rfloor$ t -ზე ნაკლები ან ტოლი უდიდესი მთელი რიცხვია).

თუ დავუშვებთ, რომ γ_i სიდიდეები დამოუკიდებელია. აღმოჩნდაა, რომ (Banks, Garson and Nelson, 1996)

$$D(\hat{\rho}_j) = \frac{13h+7}{(h+1)^2} . \tag{2.32}$$

განვიხილოთ H_0 ნულოვანი ჰიპოთეზა: რომელიც უშვებს, რომ $\rho_j = 0$. ცნობილია, რომ $\mathrm{E}(\hat{\rho}_j) = 0$. შევთანხმდეთ, რომ n საკმარისად დიდია. მაშინ

$$A_j = \frac{\hat{\rho}_j - \mathsf{E}\hat{\rho}_j}{\sqrt{D(\hat{\rho}_j)}} = \frac{\hat{\rho}_j (h+1)}{\sqrt{13h+7}} \tag{2.33}$$

სტატისტიკური კრიტერიუმი ასიმპტოტურად ნორმალურადაა განაწილებული $(A_j \xrightarrow{n\to\infty} N(0,1))$.

თუ ალტერნატიული ჰიპოთეზაა $H_1: \rho_j \neq 0$, მაშინ α მნიშვნელოვნების დონით უარვყოფთ ნულოვან ჰიპოთეზას, თუ $|A_j| > z_{1-\alpha/2}$. შევნიშნოთ, რომ ეს კრიტერიუმი უნდა შემოწმდეს ყოველი $j=1,2,3,\ldots,l$ -სთვის (საზოგადოდ აუცილებლად $l \geq 3$). გენერატორის მიერ ტესტი ჩაითვლება დადებითად გავლილი, თუ ნულოვანი ჰიპოთეზა სამართლიანია ყოველი $j=1,2,3,\ldots,l$ -სთვის.

მაგალითი. ჩვენ შევამოწმეთ IV ტესტზე წინა მაგალითებში განხილული გენერატორი, ოღონდ 5-10 ნაკადებისთვის, როცა $n=5000$. ჩავთვალეთ, რომ $l=6$. ანუ მე-5 ნაკადისათვის შევამოწმეთ კორელაცია 1 ინტერვალით. მე-6 ნაკადისთვის შევამოწმეთ კორელაცია 2 ინტერვალით და ა.შ. A_j $(j=1,2,\ldots,6)$ კრიტერიუმების მნიშვნელობები შემდეგია: 0.90, -1.03, -1.12, -1.32, 0.36, 0.76. არც ერთი 0-სგან არსებითად არ განსხვავდება $N(0,1)$

განაწილებასთან მიმართებაში. ასე რომ პირველი 5000 გენერირებული მნიშვნელობა არ ავლენს ავტოკორელაციას აღნიშნულ ინტერვალებში.

შენიშვნის სახით ვიტყვით, რომ j ინტერვალის შერჩევითი კორელაცია დავითვალეთ $\gamma_1, \gamma_{1+j}, \gamma_{1+2j}$, და ა.შ. სიდიდეებიდან. ასევე შეგვეძლო დაგვეთვალა $\gamma_2, \gamma_{2+j}, \gamma_{2+2j}$, და ა.შ. სიდიდეებიდანაც.

ბოლოს მაგალითის სახით მოვიყვანთ ცნობილ RANDU გენერატორს [1]

$$z_i = 65\,539\, z_{i-1} \pmod{2^{31}}$$

წკგ-ის ემპირიული ტესტირების *ოთხივე* ტესტით, როცა $z_0 = 123\,456\,789$. მივიღეთ შემდეგი შეფასებები:

I ტესტი: χ^2-კრიტერიუმი – $\chi^2 = 42020.0$.

II ტესტი: სერიების კრიტერიუმი – $\chi^2(2) = 4202.3$

$$\chi^2(3) = 16\,252.3.$$

III ტესტი: ზრდადი სერიების კრიტერიუმი – $R = 6.3$.

IV ტესტი: კორელაციის შემოწმება – ყველა A_j იყო უმნიშვნელო.

შევნიშნოთ, რომ I, III და IV ტესტების შედეგები დამაკმაყოფილებელია, რასაც ვერ ვიტყვით II კრიტერიუმისთვის: სამ განზომილებიან ჰიპერკუბში სამეულების თანაბარი განაწილება დარღვეულია $\chi^2(3) = 16\,252.3$-კრიტერიუმის დიდი მნიშვნელობის გამო. ამ გენერატორით სარგებლობა დაუშვებელია.

ემპირიული გენერატორების ერთ-ერთ პოტენციალურ ნაკლად შეიძლება ჩაითვალოს ის, რომ მათ გააჩნიათ მოქმედების *ლოკალური ხასიათი*, ანუ ემპირიული ტესტები იკვლევენ გენერირებული სიდიდეებს მხოლოდ იმ ციკლს, რომელიც გამოყენებული იქნა (იქნება) კომპიუტერულ მოდელირებაში. ამიტომ ის საკითხი, თუ როგორ მუშაობს გენერატორი ციკლის გარეთ დანარჩენ სიდიდეებზე, უცნობი რჩება. ეს ფაქტი გაბათილებული იქნება, თუ მოდელირებისთვის წინასწარ შეფასდება გენერირებისთვის აუცილებელ შემთხვევით რიცხვთა რაოდენობა.

§2.5.2. თეორიული ტესტების შესახებ

თეორიული ტესტებით სარგებლობას ის უპირატესობა აქვს, რომ საჭირო არაა შემთხვევითი რიცხვების გენერირება, ისე შეიძლება *აპრიორულად* განისაზღვროს რამდენად კარგად იმუშავებს იგი, თუ ცნობილი იქნება მისი სტრუქტურა და პარამეტრების მნიშვნელობები. რადგან ასეთი ტესტები ძალიან რთულია და მოითხოვს მრავალ და ამავე დროს რთულ მათემატიკურ გამოთვლებს, ამიტომ მათ მხოლოდ მოკლედ დავახასიათებთ. აქვე აღვნიშნავთ, რომ ემპირიული ტესტებისგან განსხვავებით ისინი გლობალურნი არიან და გენერატორებს იკვლევენ მთელი ციკლის გასწვრივ. თუმცა საკითხი, რომელი ტესტები უფრო უნდა გამოვიყენოთ ემპირიული თუ თეორიული, სადავოა, რადგან მიუხედავად თეორიული ტესტების გლობალურობისა, გაურკვეველი რჩება საკითხი, თუ რამდენად დამაკმაყოფილებლად იმუშავებს ციკლის ის ნაწილი, რომელიც "მოხვდება" მოდელირების პროცესში. ჩვენი რეკომენდაციით სტუდენტებისთვის ემპირიული ტესტებით სარგებლობა უფრო ხელმისაწვდომია და მისაღებია, რადგან მათი ეფექტური გამოყენება საკმარისია გენერატორის შეფასებისთვის.

შევნიშნოთ, რომ თეორიული და ემპირიული ტესტების სიმრავლის გამო არ არსებობს ერთიანი ხედვა, თუ რომელი ნაკრები ტესტებისა გამოვიყენოთ გენერატორების ტესტირებისთვის. მაშასადამე არ არსებობს უნივერსალური ინსტრუმენტი გენერატორების ერთმანეთთან ობიექტური შედარებისა.

§2.6. ერთი წკგ-ის C-პროგრამული კოდი

ვიხილავთ მულტიპლიკატიურ წკგ-ს პარამეტრებით: $m = m^* = 2^{31} - 1 = 2\,147\,483\,647$, $a = a_2 = 630\,360\,016$. ამ გენერატორის C-პროგრამული კოდი შეიძლება ჩამოვტვირთოთ www.mhhe.com/engcs/industrial/lawkelron/student/code.mhtml ელ. გვერდიდან.

პროგრამის ალგორითმი ეკუთვნის მარსესა და რობერტს (1983).

გთავაზობთ C ენაზე (სტანდარტი ANSI) პროგრამის ლისტინგს.

ლისტინგ 2.1-ში წარმოდგენილია მულტიპლიკატიური წკგ-ს C-კოდი. ლისტინგ 2.2-ში წარმოდგენილია თავფურცელი ფაილი (lcgrand.h), რომელიც გამომყენებელმა პროგრამაში უნდა ჩართოს #include-ინსტრუქციით, რათა გაცხადდეს ფაილი. ამ პროგრამით ჩვენ შემდგომში ვისარგებლებთ მასობრივი მომსახურეობის სისტემების სიმულაციისთვის.

ლისტინგი 2.1 (კოდი C-ზე; მულტიპლიკატიური წრფივი კონგრუენტული გენერატორი მარტივი მოდულით: $m = 2^{31} - 1$, $a = 630\,360\,016$).

```
/* მულტიპლიკატიური წკგ აქ მოყვანილი პარამეტრებით შექმნილია Marse და Roberts, 1983
ალგორითმის საფუძველზე; ხელს უწყობს მრავლობით 100 ნაკადს საწყისი მნიშვნელობებით,
საერთო ჯამში დაახლოებით 100 000 ნაკადს მნიშვნელობით. ყველგან შესავალი არგუმენტია
streem, რომელიც განსაზღვრავს საჭირო ნაკადის ნომერს. მისი ტიპია int. დასათაურების
ფაილი lcgrand.h ჩართული უნდა იყოს გამოძახებულ პროგრამაში (#include "lcgrand.h")
შემდეგი ფუნქციების სარგებლობის წინ.

გამოყენება (3 ვარიანტი):
  1. რომ მივიღოთ შემდეგი შემთხვევითი რიცხვი U(0,1) განაწილებიდან streem ნაკადიდან
     შეასრულეთ:
        u = lcgrand(streem);
     სადაც lcgrand float ტიპის ფუნქციაა. ცვლადი u float ტიპისაა, შეიცავს შემდეგ
     შემთხვევით რიცხვს.
  2. ნაკადი streem–ს საწყის რიცხვს რომ მივანიჭოთ zset საჭირო მნიშვნელობა, შეასრულეთ
        lcgrandst(zset, streem);
     სადაც lcgrandst viod ტიპის ფუნქციაა, ხოლო zset უნდა იყოს long ტიპის ცვლადი,
     რომელსაც მინიჭებული აქვს საჭირო საწყისი მნიშვნელობა 1-დან 2 147 483 646
     (ჩათვლით). გაჩუმებით საწყისი რიცხვები 100-ვე ნაკადისთვის მითითებულია კოდში.
  3. რომ გადავცეთ მიმდინარე მთელი რიცხვი მიმდევრობიდან, რომელიც გენერირდება streem
     ნაკადიდან, zget ცვლადში (long ტიპის), შეასრულეთ
        zget = lcgrandgt(streem);
     სადაც lcgrandgt – long ტიპის ფუნქციაა */

/* განვსაზღვრავთ მუდმივებს */
        #define MODLUS    2147483647
        #define MULT1       24112
        #define MULT2       26143

/* ყველა 100 ნაკადისთვის ვაყენებთ საწყის რიცხვებს */
  static long zrng[ ]=
  { 1,
1973272912, 281629770,  20006270,1280689831,2096730329,1933576050,
913566091, 246780520,1363774876, 604901985,1511192140,1259851944,
824064364, 150493284, 242708531,  75253171,1964472944,1202299975,
233217322,1911216000, 726370533, 403498145, 993232223,1103205531,
762430696,1922803170,1385516923,  76271663, 413682397, 726466604,
336157058,1432650381,1120463904, 595778810, 877722890,1046574445,
68911991,2088367019, 748545416, 622401386,2122378830, 640690903,
1774806513,2132545692,2079249579,  78130110, 852776735,1187867272,
1351423507,1645973084,1997049139, 922510944,2045512870, 898585771,
```

```
243649545,1004818771, 773686062, 403188473, 372279877,1901633463,
498067494,2087759558, 493157915, 597104727,1530940798,1814496276,
536444882,1663153658, 855503735,  67784357,1432404475, 619691088,
119025595, 880802310, 176192644,1116780070, 277854671,1366580350,
1142483975,2026948561,1053920743, 786262391,1792203830,1494667770,
1923011392,1433700034,1244184613,1147297105, 539712780,1545929719,
190641742,1645390429, 264907697, 620389253,1502074852, 927711160,
364849192,2049576050, 638580085, 547070247 };

/* ვაკეთებთ შემდეგი შემთხვევითი რიცხვის გენერირებას */
float lcgrand(int stream)
{
  long zi, lowprd. hi31;
  zi         = zrng [stream];
  lowprd  = (zi & 65535) * MULT1;
  hi 31     = (zi > >16) * MULT1 + (lowprd > >16);
  zi         = ((lowprd & 65535) - MODLUS) +
                        ((hi31 & 32767) << 16) + (hi 31 >> 15);
  if  (zi < 0)  zi + = MODLUS;
  lowprd  = (zi & 65535) * MULT2;
  hi 31     = (zi > >16) * MULT2 + (lowprd >> 16);
  zi         = ((lowprd & 65535) - MODLUS) +
               ((hi31 & 32767) < <16) + (hi31 > >15);
  if (zi < 0) zi + = MODLUS;
  zrng [stream] = zi;
  return (zi >> (7/1)) / 16777216.0;
}

void lcgrandst (long zset, int stream)
/* zset მნიშვნელობას ვანიჭებთ  stream ნაკადისთვის
   zrng – მიმდინარე მნიშვნელობად */
{
    zrng [stream] =zset;
}

long lcgrandgt (int stream)
/* stream ნაკადისთვის  zrng  მნიშვნელობის დაბრუნება */
{
    return zrng [stream];
}
```

ლისტინგი 2.2 (კოდი C-ზე; პროგრამის ფაილი C -ზე).

```
/* შემდეგი 3 გამოცხადება lcgrand – შემთხვევით რიცხვთა გენერატორისთვის და მასთან დაკა-
   ვშირებულ lcgrandst და lcgrandgt ფუნქციებისთვისაც, რომლებიც მართავენ საწყის მნიშვნე-
   ლობებს. ეს ფაილი (lcgrand.h) ჩართული უნდა იყოს ყველა პროგრამაში, სადაც ეს ფუნ-
   ქციები გამოიყენება “#include “lcgrand.h” ინსტრუქციით, სანამ ფუნქციას გამოვაცხადებთ */
  float lcgrand(int streem);
  void lcgrandst(long zset, int streem);
  long lcgrandgt(int streem);
```

§2.7. ამოცანები შემთხვევითი რიცხვების გენერატორებზე

2.7.1. $z_i = (5z_{i-1} + 3) \pmod{16}$, $z_0 = 7$ გენერატორისთვის "ხელით" გამოთვალეთ z_{500}.

2.7.2. გამოთვალეთ ყველა z_i, $i \geq 1$ სიდიდეები, რომლებიც სრულად ფარავენ შემდეგი მულტიპლიკატიური წკგ-ს ციკლს:

ა) $z_i = (11z_{i-1}) \pmod{16}$, $z_0 = 1$;

ბ) $z_i = (11z_{i-1}) \pmod{16}$, $z_0 = 2$;

გ) $z_i = (2z_{i-1}) \pmod{13}$, $z_0 = 1$;

დ) $z_i = (3z_{i-1}) \pmod{13}$, $z_0 = 1$.

შევნიშნოთ, რომ ა) და ბ) ამოცანებში m-ს აქვს 2^b ფორმა. გ) ამოცანაში გვაქვს მულტიპლიკატიური წკგ მარტივი მოდულით, რომლისთვისაც $a = 2$ პირველწარმოქმნილი ელემენტია მოდულით 13.

2.7.3. z_i მნიშვნელობების გამოთვლის გარეშე დაადგინეთ რომელ წკგ-ს გააჩნია სრული პერიოდი?

ა) $z_i = (13z_{i-1} + 13) \pmod{16}$;

ბ) $z_i = (12z_{i-1} + 13) \pmod{16}$;

გ) $z_i = (13z_{i-1} + 12) \pmod{16}$;

დ) $z_i = (z_{i-1} + 12) \pmod{16}$.

2.7.4. მე-3 ამოცანის ოთხივე ვარიანტისთვის გამოთვალეთ z_i-ს ყველა მნიშვნელობა, რომელიც ფარავს სრულ ციკლს, თუ $z_0 = 1$.

2.7.5. მარსესა და რობერტსის ალგორითმის პროგრამული კოდის (იხ. ლისტინგი 2.1) გამოყენებით, მოდიფიცირებით შეასრულეთ (აამუშავეთ) გენერატორი

$$z_i = (630\,360\,016\, z_{i-1}) \pmod{2^{31} - 1}$$

თქვენს კომპიუტერზე.

2.7.6.* მე-5 ამოცანის შესრულების შემდეგ შეადგინეთ C-პროგრამული კოდი ოთხივე ემპირიული ტესტისთვის და გაატარეთ გენერატორი ტესტირებაზე. როგორია თქვენი დასკვნები გენერატორის ვარგისიანობაზე (მნიშვნელოვნების დონეა $\alpha = 0.1$).

2.7.7. გთავაზობთ $f_1, f_2, \ldots, f_k$ სიდიდეების გამოთვლის ალგორითმს χ^2-კრიტერიუმის ტესტიდან. შეამოწმეთ ალგორითმი და შეადგინეთ სწრაფი C-პროგრამული კოდი.

1. დავნიშნოთ $f_j \leftarrow 0$, $j = 1, 2, \ldots, n$.
2. $i = 1, \ldots, n$ ციკლისთვის

   ვახდენთ γ_i-ს გენერირებას

   $J = \lceil k\gamma_i \rceil$

   $f_J \leftarrow f_J + 1$

   ციკლი 2-ის დასასრული

სადაც $\lceil t \rceil$ აღნიშნავს უმცირეს მთელ რიცხვს, რომელიც მეტია ან ტოლი t-სი.

2.7.8. აჩვენეთ, რომ შემდეგი ალგორითმი სწორად ითვლის $r_1, r_2, \ldots, r_6$ სიდიდეებს ზრდადი სერიების კრიტერიკუმში.

1. დავნიშნოთ $r_j \leftarrow 0$, $j = 1, 2, \ldots, 6$
2. γ_1 გენერირებული სიდიდისთვის $A \leftarrow \gamma_1$ და $J \leftarrow 1$

3. $i = 2,\ldots,n$ ციკლისთვის

ვახდენთ γ_i-ს გენერირებას და $B \leftarrow \gamma_i$

თუ $A \geq B$, მაშინ

$J \leftarrow \min(J,6)$

$r_J \leftarrow r_J + 1$

$J \leftarrow 1$

თუ $A < B$, მაშინ

$J \leftarrow J + 1$

დასასრული თუ $A \leftarrow B$

ციკლი 3-ის დასასრული

4. $J \leftarrow \min(J,6)$

5. $r_J \leftarrow r_J + 1$

2.7.9.* შეამოწმეთ თქვენს კომპიუტერში "ჩადგმული" შემთხვევით რიცხვთა გენერატორი (მაგალითად C-ენაზე ჩადგმული) ემპირიულ ტესტებზე. შეამოწმეთ χ^2-კრიტერიუმზე; 2 და 3 განზომილებიან სერიების კრიტერიუმზე, ზრდადი სერიების კრიტერიუმზე და კორელაციის ტესტზე 1, 2, 3, 4 და 5 ინტერვალებისთვის. $n = 32\,768$; $d = 2, 3$. თუ $d = 2$, მაშინ $k = 4$; თუ $d = 3$, მაშინ $k = 3$; $\alpha = 0.1$.

2.7.10.* მე-5 ამოცანის გენერატორით მოახდინეთ გენერირება $\gamma_1, \gamma_2, \ldots, \gamma_n$ n-სრული ციკლით. შეამოწმეთ, რომ ყველა γ_i-ების საშუალო, აღებული მხოლოდ სრული ციკლიდან, შეადგენს

$$\frac{1}{2} - \frac{1}{2m},$$

სადაც m მოდულია $(m = 2^{31} - 1)$.

2.7.11.* ხშირად პრაქტიკაში იყენებენ შემდეგ წკგ-ს (Knuth, 1998):

ა) $z_0 = 0,\ a = 3\,141\,592\,653,\ c = 2\,718\,281\,829,\ m = 2^{35}$;

ბ) $z_0 = 0,\ a = 2^7 + 1,\ c = 1,\ m = 2^{35}$;

გ) $z_0 = 47\,594\,118,\ a = 23,\ c = 0,\ m = 10^8 + 1$;

დ) $z_0 = 3\,141\,559\,265,\ a = 2^{18} + 1,\ c = 0,\ m = 2^{35}$

C-ენაზე დააპროგრამეთ ეს გენერატორები და შეამოწმეთ ისინი ოთხივე ემპირიულ ტესტზე. გააკეთეთ დასკვნები გენერატორების გამოყენებაზე ($\alpha = 0.1$).

თავი 3. შემთხვევითი სიდიდეების მოდელირება

შესავალი

მოდელირების პროცესში ხშირად გვიწევს რომელიმე შემთხვევითი ფაქტორის (ფაქტორების) შერჩევების, სიმულაციების (გენერირებული მიმდევრობის) შექმნა. ეს კი ნიშნავს გარკვეული განაწილების მქონე შემთხვევითი სიდიდის მოდელირებას (გათამაშებას).

"შემთხვევითი სიდიდეების სიმულაცია, გენერირება", გულისხმობს ხელოვნური ინსტრუმენტარიით (ალგორითმები, მეთოდები და სხვ.) მოცემული განაწილების მქონე შერჩევის მიღებას.

ეს განაწილებები შეიძლება იყოს: დისკრეტული (თანაბარი, ბერნულის, ბინომიალური, პუასონის და სხვ.), უწყვეტი (ექსპონენციალური, თანაბარი, ნორმალური, გამა-განაწილება, ერლანგის განაწილება და სხვ.) და კომბინირებული (შერეული)

$$F(x) = \alpha F_1(x) + (1-\alpha)F_2(x) \tag{3.1}$$

განაწილების ფუნქციით, სადაც $0 < \alpha < 1$ და F_1-უწყვეტი, ხოლო F_2 დისკრეტული ტიპის განაწილების ფუნქციებია. ამ თავში ჩვენ ვუშვებთ, რომ შემთხვევითი სიდიდის, რომლის მოდელირებაა შესაქმნელი, განაწილების ფუნქცია სტატისტიკური მეთოდებით უკვე განსაზღვრულია. მაგალითად მასობრივი მოვლენების მოდელებისთვის მოითხოვება მომდევნო *შეკვეთებს (მოთხოვნებს)* შორის დროის და მომსახურეობის დროის ინტერვალის *გენერირება (სიმულაცია)*, რათა უზრუნველყოფილი იქნას მოდელის დროის მიმდინარეობა. რაც შეეხება რესურსების მართვის სისტემებს, აქ აუცილებელია მოთხოვნის მოცულობის სიმულირება მისი წარმოშობის მომენტში.

როგორც შემდგომში აღმოჩნდება ნებისმიერი განაწილების მქონე შემთხვევითი სიდიდის გენერირების (სიმულაციის) წყაროს წარმოადგენს $U(0,1)$ კანონით განაწილებული დამოუკიდებელი შემთხვევითი სიდიდეები. მაშინ ცხადი ხდება, თუ *რატომ დავუთმეთ ჩვენს კურსში დიდი ადგილი შემთვევითი რიცხვების გენერატორების შესწავლას.* უმრავლესობა სიმულაციური მოდელირების პროგრამულ პაკეტებსა და პროგრამულ უზრუნველყოფაში არსებობს მოსახერხებელი, მაღალი ხარისხის შემთხვევით რიცხვთა გენერატორები, მაგრამ რამოდენიმე მათგანი (განსაკუთრებით ძველი ვერსიები) არაადექვატურია თანამედროვე მოთხოვნების დონისა. კარგი გენერატორის არ არსებობის გარეშე შეუძლებელია ნებისმიერი განაწილების მქონე შემთხვევითი სიდიდის *სწორი* გენერირება. მომავალში ვიგულისხმებთ, რომ ჩვენს ხელთაა სწორედ ასეთი გენერატორი.

არსებობს რამოდენიმე ალტერნატიული ალგორითმი, რომლითაც შესაძლებელია შემთხვევითი სიდიდის გენერირება. მათი არჩევისას საჭიროა რამოდენიმე ფაქტორის გათვალისწინება. შევეცდებით გავაშუქოთ რამოდენიმე საკითხი, რომელიც დაკავშირებულია ალგორითმის არჩევის პროცედურასთან.

პირველი ასეთი საკითხია *სიზუსტე.* გენერირების ეფექტური და ზუსტი ალგორითმები ხელმისაწვდომია ყველა ტიპის განაწილებისთვის, რაც საბოლოოდ უარს გვათქმევინებს გენერირების ისეთ მიახლოებით ალგორითმებზე, რომლებიც დაფუძნებულია ზღვარით თეორემებზე (ასეთი მეთოდია მაგალითად ნორმალური განაწილების მქონე შემთხვევითი სიდიდის გენერირება 6-დან 12-მდე შემთხვევითი რიცხვის საშუალოთი, რომელიც ცენტრალურ ზღვარით თეორემაზეა დამყარებული), სპეციალისტები შეიძლება არ დაგვეთანხმონ მიახლოებითი ალგორითმების უარის თქმაში, მაგრამ ყველაფერი დამოკიდებულია *მოთხოვნილი სიზუსტის მიღწევაზე* და რადგან არსებობენ ზუსტი მეთოდები, ჩვენც უპირატესობას ზუსტ მეთოდებს ვანიჭებთ.

მეორე ასეთი საკითხია *ეფექტურობა.* ზუსტი ალგორითმებიდან უკვე უპირატესობა მიენიჭებათ უფრო *ეფექტურ* ალგორითმებს. ე. ი. ისეთ ალგორითმებს, რომელთაც ესაჭიროებათ მეხსიერების მცირე რესურსები და გამოირჩევიან სწრაფქმედებით. ზოგიერთ

ალგორითმს ესაჭიროება მნიშვნელოვანი რაოდენობა მუდმივების დამახსოვრება ან დიდი მოცულობის მქონე ცხრილის დამახსოვრება, რომელიც შეიძლება უკვე პრობლემატური აღმოჩნდეს შესრულებაში. რაც შეეხება ალგორითმის შესრულების დროს, აქ ორი ფაქტორი არსებობს: ჯერ ერთი საჭიროა ყოველი შემთვევითი სიდიდის გენერირება გარკვეული მცირე დროის განმავლობაში. ამას ეწოდება *ზღვრული შესრულების დრო*; და მეორე: ზოგიერთ ალგორითმს ესაჭიროება წინასწარი გამოთვლები, რათა განისაზღვროს გარკვეული მუდმივები და ცხრილები. დროს, რომელიც საჭიროა წინასწარი გამოთვლებისთვის *მუშაობისთვის მომზადების დრო* ეწოდება. რადგან მოდელირებისას ვღებულობთ დიდი რაოდენობა გენერირებულ სიდიდეებს, ამიტომ *ზღვრული შესრულების* დროს *მეტი მნიშვნელობა ენიჭება.*

მესამე ასეთი საკითხია *ალგორითმის სირთულე.* უნდა განისაზღვროს იქნება თუ არა სარგებლიანი ისეთი ალგორითმის გამოყენება, რომლის სტრუქტურა რთულია და მისი აღქმისა და რეალიზაციისთვის მოითხოვება დამატებითი ძალისხმევა?

მეოთხე ასეთი საკითხია *ალგორითმის ტექნიკური გადაწყვეტა.* უპირველეს ყოვლისა ზოგიერთი ალგორითმი ემყარება გენერირების ისეთ წყაროს, რომელიც არაა $U(0,1)$ კანონით განაწილებული. ეს კი არაა სასურველი სხვა ალგორითმებთან მიმართებაში. გარდა ამისა ზოგიერთი ალგორითმი ეფექტურია მისი პარამეტრების გარკვეული მნიშვნელობებისთვის, ხოლო გარკვეული მნიშვნელობებისთვის კი მოითხოვს დროისა და მეხსიერების დიდ დანახარჯებს. რა საკვირველია ესეც არაა სასურველი სხვა ალგორითმებთან მიმართებაში. ჩვენ უპირატესობას ვანიჭებთ ისეთ ალგორითმებს, რომლებიც ეფექტურია მათი *პარამეტრების დასაშვები ნებისმიერი მნიშვნელობებისთვის.* ამ თვისებას ალგორითმის *მდგრადობა ეწოდება.*

მეხუთე ასეთი საკითხია *დისპერსიის დაწევა გამოთვლებში.* როგორც სტატისტიკის კურსიდანაა ცნობილი, პარამეტრის შეაფასების ხარისხი დამოკიდებულია შემფასებლის დისპერსიაზე. რაც ნაკლებია დისპერსია მით ნაკლებია რისკი დაბალი სიზუსტის მიღებაზე გენერირების ალგორითმებში. ცხადია ის ალგორითმია ეფექტური, რომლის შემფასებლებსაც ნაკლები დისპერსია გააჩნია.

შემთხვევითი სიდიდეების მოდელირებაზე არსებობს უამრავი ლიტერატურა. ძირითადი მეთოდები და ალგორითმები კი დაფარულია კურსის ლიტერატურით.

§3.1. შემთხვევითი სიდიდეების გენერირების ზოგადი მიდგომა

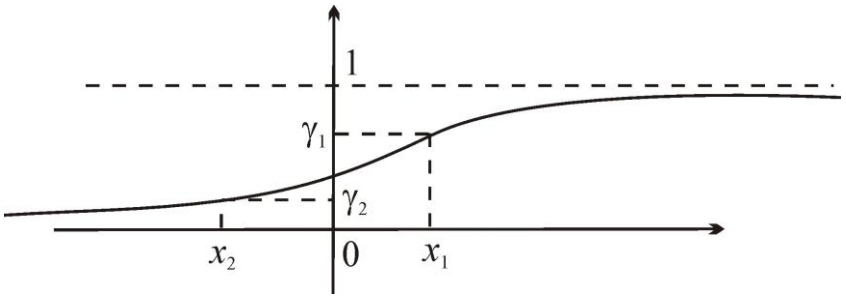
§3.1.1 შებრუნებული გარდაქმნა

ვთქვათ აუცილებელია $F(x)$ -უწყვეტი განაწილების მქონე X შემთხვევითი სიდიდის გენერირება. ცნობილია, რომ $0 \le F(x) \le 1$და დავუშვათ რომ ის მკაცრად ზრდადია (ე. ი. თუ $x_1 < x_2$, მაშინ $0 \le F(x_1) < F(x_2) \le 1$). რადგან F -უწყვეტი ფუნქციაა, მაშინ არსებობს მისი შებრუნებული ფუნქცია F^{-1}. X შემთხვევითი სიდიდის გენერირების ალგორითმი მიღებული *შებრუნებული გარდაქმნის* მეთოდით ასეთია:

1. სიმულაციით, გენერირებით მივიღოთ $\gamma \sim U(0,1)$ შემთხვევითი რიცხვი;
2. დავაბრუნოთ $X := F^{-1}(\gamma)$.

შევნიშნოთ, რომ $F^{-1}(\gamma)$ ყოველთვის იარსებებს, რადგან $0 \le \gamma \le 1$ და F^{-1}-ფუნქციის განსაზღვრის არეა $[0;1]$.

ნახ. 3.1-ზე გრაფიკულად წარმოდგენილია შებრუნებული გარდაქმნის ალგორითმი.



ნახ. 3.1

შენიშვნა. თუ შემთხვევით რიცხვთა გენერატორი გვაძლევს $\gamma_1, \gamma_2, \ldots, \gamma_n$ - შერჩევას, მაშინ შებრუნებული გარდაქმნის მეთოდი მოგვცემს $F(x)$ განაწილების ფუნქციის მქონე გენერირებული n განზომილებიან შერჩევას. ამისთვის უნდა ვაჩვენოთ, რომ $X = F^{-1}(\gamma)$ გარდაქმნით მიღებულ X შემთხვევით სიდიდეს გააჩნია $F(x)$-განაწილების ფუნქცია:

$$\Pi(X \le x) = \Pi(F^{-1}(\gamma) \le x) = \Pi(\gamma \le F(x)) = F(x).$$

მაგალითი 3.1. ვთქვათ X ექსპონენციალურად განაწილებული შემთხვევითი სიდიდეა λ-ინტენსივობის პარამეტრით $(\lambda > 0)$:

$$F(x) = \begin{cases} 1 - e^{-\lambda x}, & \text{თუ} \quad x \ge 0, \\ 0, & \text{თუ} \quad x \le 0. \end{cases}$$

$F(x) = \gamma \quad (0 < \gamma < 1)$ განტოლების ამოხსნისთვის (F -ის შებრუნებული ფუნქციის მიღებისთვის) გვაქვს:

$$1 - e^{-\lambda x} = \gamma,$$

საიდანაც $x = F^{-1}(\gamma) = -\frac{1}{\lambda}\ln(1-\gamma)$. მაშასადამე X -ექსპონენციალური სიდიდის გენერირებისთვის ჯერ გენერატორით მივიღებთ γ შემთხვევით რიცხვს, ხოლო შემდეგ

$$x = -\frac{1}{\lambda}\ln\gamma$$

ფორმულით უკვე გენერირდება X სიდიდე (აქ $1-\gamma$ შევცვალეთ γ-თი. რადგან ორივე თანაბარი განაწილებისაა $(0,1)$ -ზე). მართალია მოვიგეთ გამოკლების ოპერაციაში, მაგრამ დარწმუნებული უნდა ვიყოთ, რომ ეს შეცვლა კორელაციას არ წარმოშობს გენერირებაში!

შებრუნებული გარდაქმნის მეთოდი ასევე გამოყენებული შეიძლება იყოს, თუ შემთხვევითი სიდიდე დისკრეტულია

$$\begin{pmatrix} x_1, & x_2, & \ldots \\ p(x_1), & p(x_2), & \ldots \end{pmatrix} \tag{3.2}$$

განაწილების კანონით $p(x_i) = \Pi(X = x_i)$. ამ შემთხვევაში განაწილების ფუნქცია ასე ჩაიწერება:

$$F(x) = \Pi(X \le x) = \sum_{x_i \le x} p(x_i). \tag{3.3}$$

თუ დავუშვებთ, რომ $x_1 < x_2 < \cdots$, მაშინ *შებრუნებული გარდაქმნის* მეთოდით მიღებული ალგორითმი ასე ჩაიწერება:

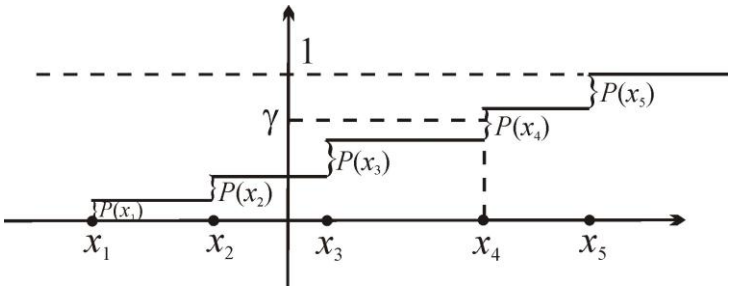
1. სიმულაციით, გენერირებით მივიღოთ $\gamma \sim U(0,1)$ შემთხვევითი რიცხვი;
2. გავნსაზღვროთ დადებითი უმცირესი მთელი რიცხვი I, რომლისთვისაც $\gamma \le F(x_I)$;
3. დავაბრუნოთ $X := x_I$.

რომ დავრწმუნდეთ ჩამოყალიბებული ალგორითმის სამართლიანობაში, აუცილებელია ვაჩვენოთ, რომ $\Pi(X = x_I) = p(x_I)$, ყველა $I = 1,2,\ldots$, მნიშვნელობებისთვის. $I = 1$-თვის მაშინ და მხოლოდ მაშინ სრულდება $X = x_1$ ტოლობა, როცა $\gamma \le F(x_1) = p(x_1)$, თანაც $\{x_I\}$ სიდიდეები დალაგებულია ზრდადობით. რადგან $\gamma \sim U(0,1)$, ამიტომ $\Pi(X = x_1) = p(x_1)$. $I \ge 2$ -თვის ალგორითმი ადგენს $X = x_I$ ტოლობას მაშინ და მხოლოდ მაშინ, როდესაც $F(x_{I-1}) < \gamma \le F(x_I)$, რადგან ალგორითმის მიერ არჩეული I -დადებითი მთელი რიცხვია, რომლისთვისაც $\gamma \le F(x_I)$. ამავე დროს $\gamma \sim U(0,1)$ და $0 \le F(x_{I-1}) < F(x_I) \le 1$. მაშინ

$$\Pi(X = x_I) = \Pi(F(x_{I-1}) < \gamma \le F(x_I)) = F(x_I) - F(x_{I-1}) = p(x_I)$$

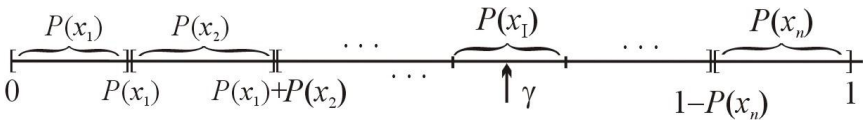
რ.დ.გ.

ნახ. 3.2 გვიჩვენებს შებრუნებული გარდაქმნის ალგორითმის მოქმედებას დისკრეტული განაწილებისთვის.



ნახ 3.2

ფაქტიურად $[0,1]$ სეგმენტი იყოფა თანაუკვეთი $\{p(x_i)\}_i$ სიგრძის ტოლი ინტერვალების გაერთიანებად:



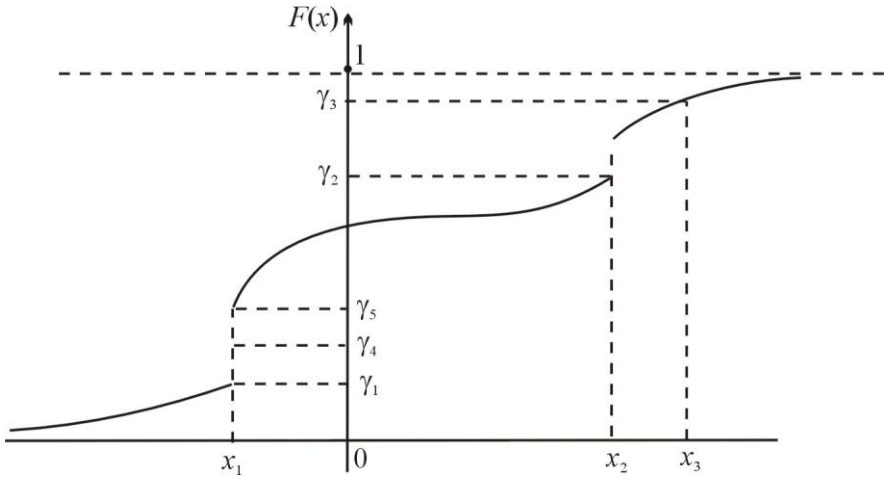
ნახ. 3.3

თუ γ "ჩავარდება" I -ურ ინტერვალში, მაშინ ჩაითვლება, რომ გენერირდა $X = x_I$ სიდიდე. ალგორითმის ეფექტურობა დამოკიდებულია იმაზე, თუ როგორ ვეძებთ იმ ინტერვალს, რომელშიც აღმოჩნდება γ. პირდაპირი მეთოდია დავიწყოთ შედარების მოქმედება მარცხენა ან მარჯვენა ბოლოდან და მიმდევრობით გავყვეთ, თუ რომელ ინტერვალში ჩავარდება γ. ე. ი. შედარებათა რიცხვი ალგორითმში დამოკიდებულია γ-სა და $\{p(x_i), i = 1,2,\ldots\}$ რიცხვებზე.

შემოგთავაზებთ ინტერვალის ძიების გაუმჯობესებულ მეთოდს (Knuth 1998): ჯერ ვამოწმებთ აღმოჩნდება თუ არა γ უდიდესი სიგრძის შუალედში (ყველაზე ალბათური შემთხვევა). თუ ეს არ მოხდა, მაშინ ვამოწმებთ აღმოჩნდება თუ არა γ მეორე სიდიდით უდიდესი სიგრძის შუალედში და ა.შ. ფორმალურად ეს ალგორითმი ასე შეიძლება ჩაიწეროს. ვთქვათ $p(x_{i_1}) \ge p(x_{i_2}) \ge \cdots \ge p(x_{i_n})$. მაშინ შედარებათა საშუალო მოსალოდნელი მნიშვნელობა ალგორითმისთვის ასე ჩაიწერება (სასრული შემთხვევითი სიდიდისთვის):

$$\sum_{k=1}^{n-1} k \cdot p(x_{i_k}) \qquad (3.4)$$

(შენიშვნა: ბოლოს n -ური შედარება აღარ ტარდება!), რომელიც კლებადობით დალაგებული ალბათობებისთვის უმცირესია.



ნახ. 3.4

დისკრეტული და უწყვეტი განაწილების გენერირების შებრუნებული გარდაქმნის ფორმულები შეიძლება გაერთიანდეს შემდეგ წარმოდგენაში. თუ $\gamma \sim U(0,1)$, მაშინ

$$X := \min\{x \in \mathrm{P} / F(x) \geq \gamma\}. \tag{3.5}$$

(3.5) ფორმულა სამართლიანია (3.1) ტიპის შერეული განაწილებისთვისაც. ნახ. 3.4 წარმოადგენს სწორედ შერეული განაწილების გენერირებას შებრუნებული გარდაქმნის მეთოდით.

ამ მეთოდს გააჩნია დადებითი და უარყოფითი თვისებები. მაგრამ მათ აქ არ შევეხებით. ჩვენი ინტერესია კონკრეტული განაწილებების *გენერირების ეფექტური ალგორითმების* აგება.

§3.1.2. კომპოზიციის მეთოდი

თუ X შემთხვევითი სიდიდის განაწილების ფუნქცია წარმოიდგენება, როგორც ამოზნექილი ჯამი

$$\Phi(x) = \sum_{k=1}^{N} p_j \Phi_j(x) \tag{3.6}$$

სადაც $0 < p_j < 1$, $\sum_{j=1}^{N} p_j = 1$ და ყოველი $\Phi_j(x)$, $j = 1,2,\ldots,N$ ფუნქცია განაწილების ფუნქციაა, რომლიდანაც შესაბამისი შერჩევის გენერირება მარტივია შებრუნებული გარდაქმნის მეთოდით.

ასეთი განაწილებით შერჩევის გენერირების ზოგადი ალგორითმი ასეთია:

1. დავაბრუნოთ გენერირებით J მთელი რიცხვი დისრეტული განაწილებიდან
$$\begin{pmatrix} 1 & 2 & \ldots & N \\ p_1 & p_2 & \ldots & p_N \end{pmatrix}$$
2. დავაბრუნოთ X შემთხვევითი სიდიდის მნიშვნელობა F_J განაწილებით შებრუნებული გარდაქმნის ალგორითმით.

ვაჩვენოთ, რომ მოცემული ალგორითმით გენერირებულ X შემთხვევით სიდიდეს გააჩნია $\Phi(x)$-განაწილების ფუნქცია:

$$\Pi(X \leq x) = \sum_{j=1}^{N} \Pi(X \leq x / J = j) \cdot \Pi(J = j) = \sum_{j=1}^{N} \Phi_j(x) \cdot p_j = \Phi(x).$$

მაგალითი 3.2. მივიღოთ ორმაგი ექსპონენციალური X განაწილების (ლაპლასის განაწილების) სიმულაცია კომპოზიციის მეთოდით. ლაპლასის განაწილების სიმკვრივეა

$$f(x) = 0{,}5e^{|x|} = 0{,}5e^{x} I_{[0,\infty)}(x) + 0{,}5e^{-x} I_{(-\infty,0)}(x),$$

შადაც I სიმრავლის ინდიკატორია.

მაშინ $F(x) = \Pi(X \le x) = \int_{-\infty}^{x} 0{,}5e^{t} I_{(-\infty,0]}(t)dt + \int_{-\infty}^{x} 0{,}5e^{-t} I_{(0,+\infty)}(t)dt$. ე.ი.

$$F(x) = \frac{1}{2}F_1(x) + \frac{1}{2}F_2(x),$$

სადაც $F_1(x)$ და $F_2(x)$ ექსპონენციალური განაწილებანია:

$$F_1(x) = \begin{cases} e^{x}, & x \le 0 \\ 1, & x > 0 \end{cases}, \qquad F_2(x) = \begin{cases} 0, & x < 0 \\ 1 - e^{-x}, & x \ge 0 \end{cases}.$$

ზემოთ წარმოდგენილი ალგორითმში $\begin{pmatrix} 1 & 2 \\ 1/2 & 1/2 \end{pmatrix}$ განაწილებით გენერირდება F_1 ან F_2 განაწილებანი და შესაბამისად X-ის გენერირებული მნიშვნელობაც მიღებული იქნება (დააპროგრამირეთ C-ენაზე კომპოზიციის მეთოდი და მაგალითისთვის აიღეთ ლაპლასის განაწილება).

3.1.3. ნახვევის მეთოდი

ვთქვათ X სიდიდე, რომლის სიმულაციაა შესაქმნელი, წარმოადგენს ერთნაირად განაწილებული და დამოუკიდებული $Y_1, Y_2, \ldots, Y_m$ შემთხვევითი სიდიდეების ჯამს:

$$X = Y_1 + Y_2 + \cdots + Y_m \tag{3.7}$$

ალგორითმს, რომელიც ჯამური სიდიდეების გენერირებას აკეთებს, *ნახვევი* ეწოდება. ვთქვათ X სიდიდის განაწილების ფუნქციაა $F(x)$, ხოლო Y_j-სიდიდის განაწილების ფუნქცია კი $G(x)$.

X სიდიდის გენერირების ალგორითმი ასეთია:

1. მოვახდინოთ m-ცალი ერთნაირად განაწილებული დამოუკიდებელი $Y_1, Y_2, \ldots, Y_m$ სიდიდეების გენერირება $G(x)$ განაწილებიდან;
2. დავაბრუნოთ X სიდიდის მნიშვნელობა $X := Y_1 + Y_2 + \cdots + Y_m$.

ალგორითმის სამართლიანობის მტკიცებულება ელემენტარულია. შევნიშნოთ, რომ ამ მეთოდი-ალგორითმით სარგებლობა ეფექტურია, თუ მარტივად და სწორად გენერირდება Y_i-სიდიდე.

მაგალითი 3.3. χ^2 *-განაწილების გენერირება.* ვთვათ $Y_1, Y_2, \ldots, Y_m$ -დამოუკიდებელი სტანდარტული ნორმალური შემთხვეითი სიდიდეებია $(Y_i \sim N(0;1))$. როგორც ცნობილია $X = Y_1^2 + Y_2^2 + \cdots + Y_m^2$ განაწილებულია χ^2 კანონით, m-თავისუფლების ხარისხით. მივიღოთ $Y_1, Y_2, \ldots, Y_m \sim N(0,1)$ გენერირებული სიდიდეები (ნორმალური განაწილების გენერატორზე ვისაუბრებთ შემდეგ პარაგრაფში). შემდეგ კი X-ის გენერირება მოხდება მისი განმარტების ფორმულით.

მაგალითი 3.4 t *-სტიუდენტის განაწილების გენერირება.* ვთქვათ $Y_i \sim N(0;1)$ და $X = \chi^2(m)$ დამოუკიდებელი შემთხვევითი სიდიდეებია, მაშინ $t = \frac{Y}{\sqrt{X}}\sqrt{m}$ -სტიუდენტის განაწილებაა m-თავისუფლების ხარისხით. მისი გენერირებისთვის აუცილებელია Y-სა და X-ის (მაგალითი 3.1-ის მიხედვით) გენერირება.

§3.2. კონკრეტული უწყვეტი შემთხვევითი სიდიდეების გენერირება

ამ პარაგრაფში გავეცნობით რამოდენიმე ცნობილი უწყვეტი განაწილების გენერირების ეფექტურ ალგორითმს.

3.2.1. თანაბარი განაწილება

თუ $X \sim U(a;b)$ – რაიმე $(a;b)$ ინტერვალზე თანაბრად განაწილებული შემთხვევითი სიდიდეა, მაშინ შებრუნებული გარდაქმნის განტოლებაა:

$$x = F^{-1}(\gamma) = a + (b-a)\gamma \quad . \tag{3.8}$$

მაშინ გენერირების ალგორითმი ასეთია:

1. მივიღეთ $\gamma \sim U(0,1)$-ს გენერირება;
2. დავაბრუნოთ X-ის მნიშვნელობა: $X := a + (b-a)\gamma$.

3.2.2. ექსპონენციალური განაწილება

ექსპონენციალური განწილების გენერირება უკვე განვიხილეთ § 3.1-ის 3.1 მაგალითში.

3.2.3. ერლანგის m-რიგის განაწილება

ერლანგის m-რიგის X შემთხვევითი სიდიდე განიმარტება ჯამით

$$X = Y_1 + Y_2 + \cdots + Y_m,$$

სადაც Y_i, $i = 1,2,\ldots,m$ დამოუკიდებელი და ერთნაირად ექსპონენციალური კანონით განაწილებული შემთხვევითი სიდიდეებია λ-პარამეტრით. შებრუნებული გარდაქმნის მეთოდით Y_i გენერირდება ფორმულით $Y_i = -\frac{1}{\lambda}\ln\gamma_i$. ხოლო Y_i სიდიდეებიდან ნახვევის ალგორითმით კი ვღებულობთ X სიდიდის გენერირებას ფორმულით:

$$X = -\frac{1}{\lambda m} \cdot \ln\left[\prod_{i=1}^{m} \gamma_i\right]. \tag{3.9}$$

გენერირების ალგორთმი ასე წარმოიდგინება.

1. მოვახდინოთ m რაოდენობა $\gamma_1, \gamma_2, \ldots, \gamma_m$ შემთხვევითი რიცხვების გენერირება;
2. დავაბრუნოთ X სიდიდე ფორმულით:

$$X = -\frac{1}{\lambda m} \cdot \ln\left[\prod_{i=1}^{m} \gamma_i\right].$$

ეს ალგორითმი წარმოადგენს შებრუნებული გარდაქმნისა და კომპოზიციის ალგორითმების კომბინაციას.

3.2.4. ნორმალური განაწილება

წინასწარ შევნიშნოთ, რომ თუ გენერირდება $X \sim N(0,1)$-სტანდარტული ნორმალური შემთხვევითი სიდიდე, მაშინ აქედან უკვე გენერირდება $Y \sim N(\mu, \sigma^2)$ ნორმალური სიდიდე ფორმულით $Y = \sigma X + \mu$. ამიტომ შემოვისაზღვრებით მხოლოდ სტანდარტული ნორმალური შემთხვევითი სიდიდის (სწშს) *გენერირებით გენერირების ბოკსისა და მიულერის* (Box and Miuller, 1958) ცნობილი ალგორითმი ძველია, მაგრამ გამოყენების მიზნით საკმაოდ არაეფექტური:

1. გენერირდეს γ_1 და γ_2 დამოკიდებული შემთხვევითი რიცხვები;

2. $X_1 := \sqrt{-2\ln\gamma_1}\cdot\cos(2\pi\gamma_2)$ და $X_2 := \sqrt{-2\ln\gamma_1}\cdot\sin(2\pi\gamma_2)$ დამოუკიდებელი სნშს-ების რეალიზაციებია.

მეთოდის სისუსტე მდგომარეობს იმაში, რომ შემდეგი გენერაცია მოითხოვს დამოუკიდებელ γ -ს მნიშვნელობებს. ეს კი შემთხვევითი რიცხვების გენერირების ერთ ნაკადში არ არსებობს.

პრაქტიკულად აღნიშნულ ალგორითმს არ იყენებენ, რამეთუ შემდეგი ალგორითმი გაცილებით ეფექტურია. *მარსალისა და ბრეის* (Marsaglia and Bray, 1964) *პოლარულ კოორდინატთა მეთოდი* დაფუძნებულია ნორმალური განაწილების სპეციალური თვისებაზე:

1. გენერირდეს γ_1 და γ_2 დამოუკიდებელი შემთვევითი რიცხვები.
2. დავაბრუნოთ $\eta_i = 2\gamma_i - 1,\ i = 1,2$ და $\xi = \eta_1^2 + \eta_2^2$;
3. თუ $\xi > 1$, მაშინ დავუბრუნდეთ 1 ბიჯს, თუ არა და $Y := \sqrt{(-2\ln\xi)/\xi}$;
4. დავაბრუნოთ $X_1 := \eta_1 Y$ და $X_2 := \eta_2 Y$. მაშინ X_1 და X_2 დამოუკიდებელი $N(0,1)$ -სნშს-ების რეალიზაციებია.

ამ ალგორითმის სიკეთე ისაა, რომ აქ აღარაა საჭირო ტრიგონომეტრიული ფუნქციების გამოთვლა, რომელსაც დიდი დრო მიაქვს. მისი ეფექტურობა გარანტიებულია.

§3.3. კონკრეტული დისკრეტული შემთხვევითი სიდიდეების გენერირება

ამ პარაგრაფში განვიხლავთ სხვადასხვა მნიშვნელოვანი დისკრეტული განაწილებების გენერირების ეფექტურ ალგორითმებს, რომლებიც წარმატებით გამოიყენებიან სიმულაციურ მოდელირებაში.

ბევრი მეთოდი, რომელსაც აქ აღვწერთ, წარმოადგენს შებრუნებული გარდაქმნის ეფექტურ ალგორითმებს. ამ მეთოდის გარდა დისკრეტული განაწილებების სიმულაციაში გამოიყენება *ჩანაცვლების მეთოდი*, რომელიც უოლკერს (Walker, 1977) ეკუთვნის.

3.3.1. ბერნულის განაწილება- $Be(p)$

შებრუნებული გარდაქმნის მეთოდის ბაზაზე წარმოგიდგენთ ბერნულის განაწილების გენერირების ალგორთმს:

1. გენერირებით მივიღოთ γ -შემთვევითი რიცხვი;
2. თუ $\gamma \le p$, მაშინ $X = 1$, თუ არა და $X = 0$.

3.3.2. დისკრეტული თანაბარი განაწილება

თუ X -თანაბრად განაწილებულია $[m;n]$ -მთელი რიცხვების ინტერვალში, მაშინ წარმოგიდგენთ მისი გენერირების ალგორითმს:

1. გენერირებით მივიღოთ γ -შემთხვევითი რიცხვი;
2. დავაბრუნოთ $X := m + \lfloor (n-m+1)\gamma \rfloor$.

შენიშვნა. ძიება მე-2 ბიჯზე საჭირო არაა, წინასწარ დაიმახსოვრება $n-m+1$.

3.3.3. ბინომიალური განაწილება- $Bi(n,k;p)$

ვთქვათ $X \sim Bi(n,k;p)$ -ბინომიალური განაწილებაა (n -დამოუკიდებელი ცდების რაოდენობა; k "წარმატებათა" რიცხვია, $k = 0,1,\ldots,n$; p წარატების ალბათობაა ერთ ცდაში, $0 < p < 1$).

შევნიშნოთ, რომ $X = Y_1 + Y_2 + \cdots + Y_n$, სადაც $Y_1, Y_2, \ldots, Y_m$ - ბერნულის დამოუკიდებელი შემთხვევითი სიდიდეებია. გენერირების ალგორითმი ასე გამოიყურება:

1. მივიღოთ გენერირებით $Y_1, Y_2, \ldots, Y_m$ დამოუკიდებელი ბერნულის განაწილების შერჩევა;
2. დავაბრუნოთ $X := Y_1 + Y_2 + \cdots + Y_n$.

3.3.4. პუასონის განაწილება-$\Pi(\lambda)$

$X \sim \begin{pmatrix} 0, & 1, & 2, & \ldots \\ p_0 & p_1 & p_2 & \ldots \end{pmatrix}$ (სადაც $p_k = \frac{e^{-\lambda}\lambda^k}{k!}$, $k = 0,1,\ldots; \lambda > 0$) -პუასონის განაწილების გენერირებისთვის გთავაზობთ ალგორითმს:

1. განვმარტოთ $a := e^{-\lambda}$, $b = 1$ და $i = 0$;
2. გენერირებით მივიღოთ $\gamma_{i+1} \sim U(0,1)$ -შემთხვევითი რიცხვი და დავაბრუნოთ $b := b \cdot \gamma_{i+1}$;
3. თუ $b < a$, მაშინ დავაბრუნოთ $X := i$, თუ არა გადადი მე-4 ბიჯზე;
4. დავაბრუნოთ $i := i + 1$ და გადადი მე-2 ბიჯზე. ალგორითმი სამართლიანი იქნება მაშინ, როცა

$$X = i \Leftrightarrow \sum_{j=1}^{i} Y_j \le 1 \le \sum_{j=1}^{i+1} Y_j,$$

სადაც $Y_i = -\frac{1}{\lambda}\ln\gamma_i \sim Exp(\lambda)$ -ექსპონენციალური განაწილებაა, ხოლო Y_i სიდიდეები დამოუკიდებლებია. ანუ $X = \max\left\{ i : \sum_{j=1}^{i} Y_j \le 1 \right\}$.

სამწუხაროდ ეს ალგორითმი იწყებს ნელა მუშაობას $\lambda > 0$ პარამეტრის გაზრდით, რადგან $a = e^{-\lambda}$ სიდიდე მცირდება და ალგორითმის მე-2 ბიჯზე მეტი მოქმედებების ჩატარება გვიწევს. შემუშავებულია უფრო სწრაფი და ეფექტური გენერატორები (Devroye, 1981; Schmeiser, 1981 და ა. შ.) ერთ-ერთი ასეთი ალგორითმი ეფუძნება ფაქტს, რომ

$$X = \min\left\{ i : \prod_{j=1}^{i} \gamma_j < e^{-\lambda} \right\}$$

სიდიდე განაწილებულია პუასონის კანონით.

მოვიყვანოთ ალგორითმი:

1. განვმარტოთ $a := e^{-\lambda}$, $b := 1$, $i = 1$;
2. გენერირებით მივიღოთ γ_i დავაბრუნოთ $b = b \cdot \gamma_i$. თუ $b < a$, მაშინ $X := i$ თუ არა და გადადი 3-ზე;
3. $i := i + 1$ და გადადი მე-2-ზე (აჩვენეთ მოყვანილი ალგორითმების ექვივალენტობა).

3.3.5. პუასონის პროცესის მოდელირება

პუასონის (სტაციონალური) პროცესის ($\lambda > 0$ პარამეტრით) მოდელირება ტოლფასია $A_i = t_i - t_{i-1}$-პუასონის ხდომილების მოდელირებისა, სადაც t_{i-1} და t_i მომდევნო ხდომილებების დადგომებს შორის დროის მომენტებია: აქედან

$$t_i = t_{i-1} + A_i.$$

t_i-პუასონის ნაკადის მომენტის სიმულაციის ალგორითმი ასეთია:

1. გენერირებით მივიღოთ γ-შემთხვევითი რიცხვი;
2. დავაბრუნოთ: $t_{i:} = t_{i-1} - \frac{1}{\lambda}\ln\gamma$. რეკურსია იწყება t_1-ის გამოთვლით $(t_0 = 0)$.

§3.4. ამოცანები და სავარჯიშოები შემთხვევითი სიდიდეების მოდელირებაზე (გენერირებაზე)

3.4.1. დისკრეტული შემთხვევითი სიდიდის სიმულაცია (გათამაშება)

სიმულაციის წესი 1: ვთქვათ მოცემულია რაიმე X დისკრეტული შემთხვევითი სიდიდე

X	x_1	x_2	…	x_n
Π	p_1	p_2	…	p_n

განაწილების კანონით. იმისთვის, რომ მივიღოთ X შემთხვევითი სიდიდის სიმულაცია (გათამაშება) საჭიროა

ა) დავყოთ $(0,1)$ ინტერვალი n რაოდენობა თანაუკვეთი ინტერვალებად: $\Delta_1 = (0; p_1]$,

$\Delta_2 = (p_{1;} p_1 + p_2]$, …, $\Delta_i = \left(\sum_{j=1}^{i-1} p_i; \sum_{j=1}^{i} p_i\right], \ldots, \Delta_n = \left(\sum_{j=1}^{n-1} p_i; 1\right)$.

ბ) შემთხვევითი რიცხვების გენერატორით ვირჩევთ γ-შემთხვევით რიცხვს. ქვემოთ ამოცანებში გენერატორის როლში აიღეთ შემთხვევითი რიცხვების ცხრილი (იხ. დანართი) თუ γ ეკუთვნის რაიმე Δ_i ინტერვალს, მაშინ X შემთხვევითი სიდიდის გათამაშებული (სიმულირებული) მნიშვნელობაა- x_i.

გ) სიმულაცია ტარდება ვთქვათ N-ჯერ და მიიღება X დისკრეტული შემთხვევითი სიდიდის N-განზომილებიანი შერჩევა:

$$x^{(1)}, x^{(2)}, \ldots, x^{(N)},$$

სადაც $x^{(k)}$ არის k-ური გათამაშებით მიღებული რეალიზაცია.

3.4.1.1. გაათამაშე X-დისკრეტული შემთხვევითი სიდიდის ექვსი შესაძლო მნიშვნელობა, რომლის განაწილების კანონია (ისარგებლეთ შემთხვევითი რიცხვების ცხრილით):

X	21	10	18
Π	0,22	0,17	0,61

3.4.1.2. გაათამაშე X-დისკრეტული შემთხვევითი სიდიდის 8 შესაძლო მნიშვნელობა, რომლის განაწილების კანონია (ისარგებლეთ შემთხვევითი რიცხვების ცხრილით):

X	3	8	12	23
Π	0,2	0,12	0,43	0,23

3.4.1.3. გაათამაშე 5 შესაძლო მნიშვნელობა ბერნულის სქემით, თუ სქემა შედგება 3 დამოუკიდებელი ცდისგან, რომელთაგან თითოეულში წარმატების ალბათობაა-0.4.

მითითება.

ა) შეადგინე ბერნულის განაწილება $n=3$ დამოუკიდებელი ცდისთვის, როდესაც წარმატება ალბათობაა $p=0,4$.

ბ) მოახდინეთ განაწილების სიმულაცია 5-ჯერ (ისარგებლეთ შემთხვევითი რიცხვების ცხრილით).

3.4.2. ხდომილებათა სრული სისტემის გათამაშება

განსაზღვრება. $A_1, A_2, \ldots, A_n$ -ხდომილებათა *სისტემას ეწოდება სრული*, თუ $\bigcup_{i=1}^{n} A_i = \Omega$ და $A_i \bigcap A_j = \varnothing$, თუ $i \neq j$.

სიმულაციის წესი 2. ვთქვათ $p_i = \Pi(A_i)$. ცხადია $\sum_{i=1}^{n} p_i = 1$. დავუშვათ p_i რიცხვები ცნობილია.

იმისთვის, რომ მოვახდინოთ $A_1, A_2, \ldots, A_n$ -ხდომილებათა სრული სისტემის სიმულაცია საკმარისია სიმულაციის წესი 1-ის მიხედვით მოვახდინოთ

X	1	2	…	n
Π	p_1	p_2	…	p_n

შემთხვევითი სიდიდის სიმულირება. $X = i$ მნიშვნელობის გათამაშება ნიშნავს A_i ხდომილების გათამაშებას.

შემდეგი ამოცანებისთვის ისარგებლეთ შემთხვევითი რიცხვების ცხრილით.

3.4.2.1. ხდომილებათა A_1, A_2, A_3 სრული სისტემისათვის $p_1 = 0,22$; $p_2 = 0,31$; $p_3 = 0,47$. გაათამაშე სრული სისტემის 5 შესაძლო ხდომილება.

3.4.2.2. A და B ხდომილებები თავსებადი ($A \cap B \neq \varnothing$) და დამოუკიდებელია ($\Pi(A \cap B) = \Pi(A) \cdot \Pi(B)$). $\Pi(A) = 0,7$, $\Pi(B) = 0,4$. გაათამაშე ოთხი ცდა, რომლის სრული სისტემის გენერატორები A და B ხდომილებებია.

შენიშვნა. შესაძლებელია სრული სისტემის ოთხი ხდომილება:

$A_1 = A \cap B$ და $p_1 = \Pi(A \cap B) = \Pi(A) \cdot \Pi(B) = 0,7 \cdot 0,4 = 0,28$

$A_2 = A \cap \overline{B}$ და $p_2 = \Pi(A \cap \overline{B}) = \Pi(A) \cdot \Pi(\overline{B}) = 0,7 \cdot 0,6 = 0,42$

$A_3 = \overline{A} \cap B$ და $p_3 = \cdots$

$A_4 = \overline{A} \cap \overline{B}$ და $p_4 = \cdots$

3.4.2.3. A , B და C ხდომილებები ერთობლიობაში დამოუკიდებელი და თავსებადია. $\Pi(A) = 0,5$, $\Pi(B) = 0,2$, $\Pi(C) = 0,4$. გაათამაშე ხუთი ცდა, რომლის სრული სისტემის გენერატორებია A, B და C ხდომილებები.

3.4.2.4. A და B ხდომილებები დამოკიდებული და თავსებადია. $\Pi(A) = 0,5$, $\Pi(B) = 0,6$, $\Pi(A \cap B) = 0,2$. გაათამაშე ხუთი ცდა, რომლის სრული სისტემის გენერატორებია A და B ხდომილებები.

შენიშვნა.

$A_1 = A \cap B$ და $p_1 = \Pi(A \cap B) = 0,2$

$A_2 = A \cap \overline{B}$ და $p_2 = \Pi(A \cap \overline{B}) = \Pi(A) - \Pi(A \cap B) = 0,5 - 0,2 = 0,3$

$$A_3 = \overline{A} \cap B \quad \text{და} \quad p_3 = \Pi(\overline{A} \cap B) = \Pi(B) - \Pi(A \cap B) = 0{,}6 - 0{,}2 = 0{,}4$$
$$A_4 = \overline{A} \cap \overline{B} \quad \text{და} \quad p_4 = \Pi(\overline{A} \cap \overline{B}) = 1 - (p_1 + p_2 + p_3) = 1 - 0{,}9 = 0{,}1.$$

3.4.3. უწყვეტი შემთხვევითი სიდიდის სიმულაცია (გათამაშება)

სიმულაციის წესი 3. ვთქვათ $F(x)$ რაიმე X -უწყვეტი შემთხვევითი სიდიდის განაწილების ფუნქციაა.

ა) იმისთვის, რომ გავათამაშოთ X -უწყვეტი შემთხვევითი სიდიდის შესაძლო მნიშვნელობა: $X = x \quad (x \in \mathrm{P})$, საკმარისია ამოვხსნათ

$$F(x) = \gamma$$

განტოლება, სადაც γ -შემთხვევითი რიცხვია (ამ წესს ეწოდება გარდაქმნის შებრუნებული მეთოდი)

ბ) თუ $f(t)$ X -უწყვეტი შემთხვევითი სიდიდის განაწილების სიმკვრივეა, მაშინ გათამაშების წესი გულისხმობს

$$\int_{-\infty}^{x} f(t)dt = \gamma$$

განტოლების ამოხსნას.

გ) თუ სიმულაცია ტარდება N -ჯერ, მაშინ $\gamma_1, \gamma_2, \ldots, \gamma_N$ შემთხვევითი რიცხვების, ა) ან ბ) განტოლებაში ჩასმით და ამოხსნით მივიღებთ X -უწყვეტი შემთვევითი სიდიდის N -განზომილებიან შერჩევას სიმულაციით.

3.4.3.1. იპოვე (a,b) -ზე თანაბრად განაწილებული X შემთხვევითი სიდიდის სიმულირების ფორმულა.

შენიშვნა. განაწილების ფუნქციის ანალიზური სახეა: $F(x) = \dfrac{x-a}{b-a}$, $x \in (a,b)$.

3.4.3.2. გაათამაშე $(4;14)$-ზე თანაბრად განაწილებული შემთხვევითი სიდიდის 4 შესაძლო მნიშვნელობა (ისარგებლეთ შემთხვევითი რიცხვების ცხრილით).

3.4.3.3. იპოვეთ γ -ინტენსივობის მქონე ექსპონენციური განაწილების სიმულირების ფორმულა.

შენიშვნა. განაწილების ფუნქციის ანალიზური სახეა: $F(x) = 1 - \lambda e^{-\lambda x}$, $x > 0$.

3.4.3.4. ა) გაათამაშე $f(x) = 0{,}1e^{-0{,}1x}$ $(x > 0)$ სიმკვრივით მოცემული ექსპონენციალური განაწილებული შემთხვევითი სიდიდის 5 შესაძლო მნიშვნელობა (ისარგებლეთ შემთხვევითი რიცხვების ცხრილით).

ბ) ა) ამოცანისთვის შეადგნე C-კოდი და სიმულაცია ჩაატარე 50 და 100-ჯერ. გრაფიკულად წარმოადგინე ჰისტოგრამა 5 და 10 ინტერვალისთვის.

3.4.3.5. გაათამაშე $f(x) = \begin{cases} 1 - \dfrac{x}{2}, & \text{თუ} \quad x \in (0,2) \\ 0, & \text{თუ} \quad x \notin (0,2) \end{cases}$ -სიმკვრივით მოცემული შემთხვევითი სიდიდის 4 შესაძლო მნიშვნელობა (ისარგებლეთ შემთხვევითი რიცხვების ცხრილით).

3.4.3.6. გაათამაშე $f(x)=\begin{cases}\frac{1}{2}\sin x, & \text{თუ} \quad x\in(0,\pi)\\ 0, & \text{თუ} \quad x\notin(0,\pi)\end{cases}$ -სიმკვრივით მოცემული შემთხვევითი სიდიდის 4 შესაძლო მნიშვნელობა (ისარგებლეთ შემთხვევითი რიცხვების ცხრილით).

3.4.4. ნორმალური განაწილების სიმულაციის მიახლოებითი მეთოდი

სიმულაციის წესი 4. ა) $N(0,1)$ -სტანდარტული ნორმალური კანონით განაწილებული შემთხვევითი სიდიდის სიმულირების მიახლოებითი ფორმულაა

$$x=\sum_{i=1}^{12}\gamma_i-6,$$

სადაც $\gamma_1,\gamma_2,\ldots,\gamma_{12}$-შემთხვევითი რიცხვების გენერატორიდან მიღებული მიმდევრობაა.

ბ) $x_1,x_2,\ldots,x_N$ -ნორმალური შერჩევის სიმულაციისთვის საჭიროა სიმულირების ფორმულის N -ჯერ გამოყენება შემთხვევითი რიცხვების გენერატორის მიმდევრობითი გამოყენებით. (საჭიროა $12N$ -შემთხვევითი რიცხვი).

გ) $N(\mu,\sigma)$-კანონით განაწილებული რიცხვის გათამაშება განხორციელდება

$$\{\sigma x_i+\mu\}_{i=1}^{N}$$

ფორმულით.

3.4.4.1. გაათამაშე ა) $\mu=0$, $\sigma=1$; ბ) $\mu=10$, $\sigma=2$; პარამეტრების მქონე ნორმალური შემთხვევითი სიდიდის 5 შესაძლო მნიშვნელობა (ისარგებლეთ შემთხვევითი რიცხვების ცხრილით). გ), ა) და ბ) ამოცანებისთვის შეადგინე C-კოდი და სიმულაცია ჩაატარე 50 და 100-ჯერ. თითოეულ შემთხვევაში გრაფიკულად წარმოადგინე ჰისტოგრამა 5 და 10 ინტერვალისთვის (ჰისტოგრამა შეიძლება გაკეთდეს EXCEL-ში).

3.4.5. დისკრეტული ორგანზომილებიანი შემთხვევითი ვექტორის სიმულაცია

სიმულაციის წესი 5. ვთქვათ (x,y) შემთხვევითი ვექტორია X და Y დისკრეტული კომპონენტებით.

ა) თუ X და Y დამოუკიდებლებია, მაშინ მათი განაწილების კანონიდან ისინი გათამაშდებიან და წყვილები იძლევა გათამაშებას.

ბ) თუ X და Y დამოკიდებლებია, მაშინ მოიძებნება ერთის განაწილების კანონი მეორეს პირობითი განაწილების კანონით. აქ სიმულირება ემთხვევა ა) პუნქტის წესს.

3.4.5.1. X და Y დამოუკიდებელი დისკრეტული შემთხვევითი სიდიდეების ერთობლივი განაწილების კანონია:

$X \setminus Y$	10	11	12
1	0,18	0,20	0,08
2	0,12	0,20	0,08

გაათამაშე (X,Y) შემთხვევითი ვექტორის 4 შესაძლო მნიშვნელობა (ისარგებლეთ შემთხვევითი რიცხვების ცხრილით).

3.4.5.2. X და Y დამოკიდებული დისკრეტული შემთხვევითი სიდიდეების ერთობლივი განაწილების კანონია

$X \setminus Y$	10	11	12
1	0,10		0,06
2	0,30		0,18
3	0,20		0,16

გაათამაშე (X,Y) შემთხვევითი ვექტორის 5 შესაძლო მნიშვნელობა (ისარგებლე შემთხვევითი რიცხვების ცხრილით).

შენიშვნა. იპოვე X სიდიდის განაწილების კანონი და მიღებული კანონით გაათამაშე X . იპოვე Y -ის X -ზე დამოკიდებული პირობითი განაწილება $p(Y = Y_i / X = X_j) = \frac{p(x_i, y_i)}{p(x_j)}$ ფორმულით. ამ ფორმულით გაათამაშე y სიდიდე. შეადგინე გათამაშებული წყვილი.

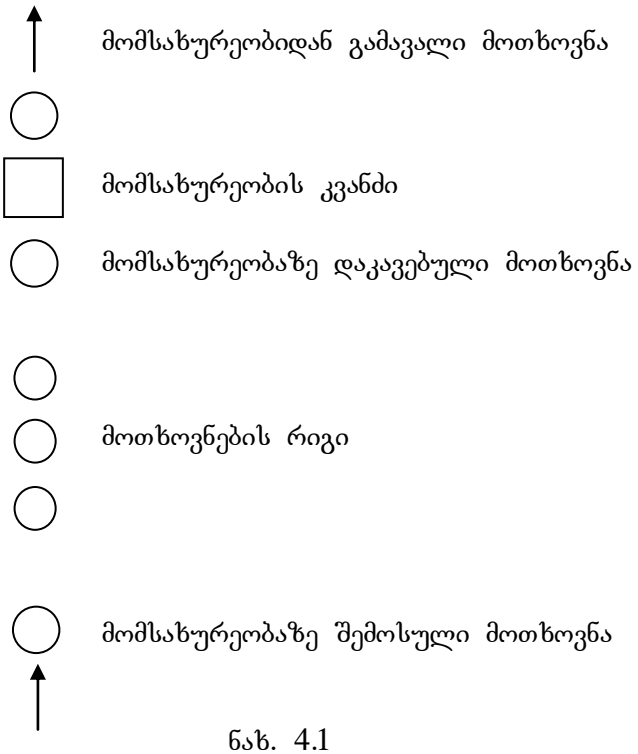
თავი 4. მასობრივი მომსახურეობის სისტემების მოდელირება (მომსახურეობის ერთი კვანძის შემთხვევა)

შესავალი

ამ თავში განვიხილავთ დისკრეტულ-ხდომილებითი სიმულაციური მოდელირების ტიპიურ მაგალითს: მასობრივი მომსახურეობის სისტემა მომსახურეობის ერთი კვანძით (მმსმეკ) (შესავალი ამ სისტემაზე წარმოდგენილი იყო თავი 1-ის მაგ. 1.1-ში). ესაა საპარიკმახერო ერთი თანამშრომლით, ბანკი ერთი მოლარით, ტელეფონების დაზიანების ბიურო მომსახურეობის ერთი კვანძით და ა.შ. ეს სისტემა იმ რთულ სისტემებთან შედარებით, რომლებიც რეალურად მოდელირდებიან, ძალიან მარტივია, თუმცა საკმაოდ მახასიათებელია მსგავსი, თუნდაც რთული სისტემების ანალიზისთვის.

§4.1. ამოცანის დასმა

განვიხილოთ მმსმეკ (ნახ. 4.1)



ნახ. 4.1

ამ სისტემაში $A_1, A_2, \ldots$ – მოთხოვნათა დადგომებს შორის დროის ინტერვალები *დამოუკიდებელი და ერთნაირად განაწილებული* შემთხვევითი სიდიდეებია. როდესაც მოთხოვნა შემოდის სისტემაში და მომსახურეობის კვანძი თავისუფალია, მომსახურეობა იწყება მყისიერად. $S_1, S_2, \ldots$ – შემდგომ მოთხოვნებზე მომსახურეობის დროთა მიმდევრობა ასევე დამოუკიდებელი და ერთნაირად განაწილებული შემთხვევითი სიდიდეებია.

თუ მოთხოვნის სისტემაში შემოსვლისას კვანძი დაკავებულია, მაშინ მოთხოვნა ჩადგება რიგში. მოთხოვნაზე მომსახურეობის დასრულების შემდეგ კვანძი მომსახურეობაზე ირჩევს მოთხოვნას რიგიდან (თუ ასეთი არსებობს) პრინციპით: "პირველი მოხვედი – პირველს მოგემსახურებიან" (FIFO – "First – In, First – Out"). ასევე დასაშვებია მომსახურეობის სხვა პრინციპებიც. მოდელირება იწყება "ცარიელი და არადაკავებული" მდგომარეობიდან, ანუ რო-

დესაც სისტემაში არაა შემოსული მოთხოვნა, ხოლო მომსახურეობის კვანძი თავისუფალია. 0 დროის მომენტიდან ჩვენ ველოდებით პირველი მოთხოვნის დადგომას. უფრო ზუსტად კი ეს მოხდება A_1 დროის ინტერვალის დასასრულს, და არა 0-ვან მომენტში (თუმცა ეს უკანასკნელი დასაშვებია გარკვეულ მოდელებში (იხ. სავარჯიშოები)). მოდელირება შეწყდება მაშინ, როდესაც რიგით n-ური მოთხოვნა იქნება მომსახურებული. შესაბამისად მოდელირების დასრულების დრო შემთხვევითია. იგი დამოკიდებულია იმ შემთხვევითი ცვლადების დაკვირვებად მნიშვნელობებზე, რომლებიც აღწერენ მოთხოვნებს შორის დროსა და მომსახურეობის დროს.

იმისთვის, რომ განვსაზღვროთ *სისტემის მუშაობის შეფასების კრიტერიუმი* (სმშკ), მივმართავთ მახასიათებლების *სამ პარამეტრს*. უპირველეს ყოვლისა უნდა ვიპოვოთ ყოველი n რაოდენობა მოთხოვნისთვის *რიგში მოთხოვნის დაკავების მოსალოდნელი საშუალო დრო*, რომელიც $d(n)$-ით აღინიშნება. სიტყვა "მოსალოდნელი" $d(n)$-ის განმარტებაში ნიშნავს იმას, რომ მოცემული სიმულაციური მოდელის გატარებისას (გავლისას), n რაოდენობა მოთხოვნის მომსახურეობისას რიგში მოთხოვნის საშუალო დაკავების დრო დამოკიდებულია იმ შემთხვევითი ცვლადების დაკვირვებად მნიშვნელობებზე, რომლებიც აღწერენ მოთხოვნების დადგომის დროსა და მისი მოსახურეობის დროს შორის სიდიდეს. სიმულაციური მოდელის *მეორედ გატარებისას* $d(n)$-ის რეალიზებული მნიშვნელობაც სხვა იქნება, რადგან რეალიზაციები უკვე სხვაა, და ა.შ. ჩვენ კი გვესაჭიროება ყოველ გატარებაზე რიგში მოთხოვნის დაკავების საშუალო მნიშვნელობა. იგი შემთხვევითი სიდიდეა, ხოლო ასეთი სიდიდეების მათემატიკური ლოდინი – რიგში მოთხოვნის დაკავების მოსალოდნელი საშუალო მნიშვნელობა კი რიცხვია – $d(n)$. ერთი გატარების შემდეგ ფიქსირდება $D_1, D_2, \ldots, D_n$ – რიგში მოთხოვნათა დაკავებაზე დროის მნიშვნელობები. $d(n)$-ის *შეფასებას* ერთ გატარებაზე ვითვლით ფორმულით:

$$\hat{d}(n) = \sum_{i=1}^{n} \frac{D_i}{n}. \qquad (4.1)$$

უნდა შევნიშნოთ, რომ როდესაც ვლაპარაკობთ დაკავების დროზე, ჩვენ არ გამოვრიცხავთ იმის შესაძლებლობას, რომ ის 0-ის ტოლი იქნება, როდესაც სისტემა იმყოფება არადაკავებულ მდგომარეობაში. ერთ-ერთი მიზეზი იმისა, რომ დაკავების ზომად ვიღებთ $D_1, D_2, \ldots, D_n$ სიდიდეების საშუალოს, არის ის, რომ მათ *არ გააჩნიათ ერთნაირი განაწილება*, ხოლო საშუალო არითმეტიკული გამოსახავს რიგში ყველა მოთხოვნის დაკავების ერთიან მახასიათებელს. აქვე აღვნიშნავთ, რომ $D_1, D_2, \ldots, D_n$ სიდიდეები *არაა დამოუკიდებელი შემთხვევითი* სიდიდეები. ასევე (4.1) ფორმულა $d(n)$-ის მხოლოდ შეფასებაა, როდესაც გამოყენებულია სიმულაციური მოდელირების *მხოლოდ ერთი გატარება*.

$d(n)$ – *რიგში მოთხოვნის დაკავების მოსალოდნელი საშუალო დრო* ინფორმაციის მატარებელია სისტემის მუშაობის კრიტერიუმის შეფასებაზე – *მოთხოვნათა უზრუნველყოფის აზრით. სისტემის მართვის აზრით* საჭირო იქნება სხვანაირი ინფორმაცია. რადგან შექმნილი უმრავლესობა სიმულაციური მოდელებისა რთულია და მათი თუნდაც ერთი გატარებისთვის არც თუ ისე ცოტა დროა საჭირო. როგორც წესი მიმართავენ სისტემის მუშაობის კრიტერიუმების მრავალ გამოსავალ შეფასებებს. ერთ-ერთ ასეთ კრიტერიუმს ჩვენს მარტივ სისტემაში წარმოადგენს *რიგში მოთხოვნათა მოსალოდნელი საშუალო მნიშვნელობა* (მომსახურეობაზე გასული მოთხოვნის ჩაუთვლელად). ის $q(n)$-ით აღინიშნება, სადაც n მიუთითებს, რომ საშუალო ითვლება იმ დროის განმავლობაში, რომელიც აუცილებელია რიგში n-ური დაკავებული მოთხოვნის დასაფიქსირებლად და რომელიც ასევე მოდელირების შეწყვეტის პირობასაც წარმოადგენს. $q(n)$ განსხვავდება $d(n)$-სგან, რადგან $q(n)$ გამოითვლება *უწყვეტი დროის გასწვრივ*, ხოლო $d(n)$ კი გამოითვლება *დისკრეტულ მოთხოვნათა გასწვრივ*. ასე რომ საჭიროა განისაზღვროს რიგში დაკავებულ მოთხოვნათა საშუალო რიცხვი დროის მიხედვით. ამ მიზნით ნებისმიერი t $(t \geq 0)$ დროის მომენტისთვის შემოსული დაკავებულ

მოთხოვნათა რიცხვი აღინიშნება $Q(t)$-თი, ხოლო დრო, რომელიც აუცილებელია რიგში n-ური დაკავებული მოთხოვნის ფიქსაციისთვის, კი – $T(n)$-ით. მაშინ დროის ნებისმიერი $t \in [0;T(n)]$ მნიშვნელობისთვის $Q(t)$ არაუარყოფითი მთელი რიცხვია. ვთქვათ p_i - მოდელირების დროის *მოსალოდნელი წილია (ნაწილია)* იმისა, როდესაც $Q(t) = i \quad (0 < p_i < 1)$. მაშინ $q(n)$-ის რაოდენობრივი განმარტებაა ფორმულა –

$$q(n) = \sum_{i=0}^{\infty} i p_i \,. \tag{4.2}$$

$q(n)$ არის *რიგში $Q(t)$-ს შესაძლო მნიშვნელობების შეწონილი ჯამი*, ამავე დროს წონითი კოეფიციენტები წარმოადგენს იმ დროის მოსალოდნელ წილს, რომელიც საჭიროა დაიხარჯოს შესაბამისი სიგრძის მომსახურეობაზე.

$q(n)$-ის შეფასებისთვის (4.2)-ში p_i იცვლება მისი შეფასებით:

$$\hat{q}(n) = \sum_{i=0}^{\infty} i \hat{p}_i \,. \tag{4.3}$$

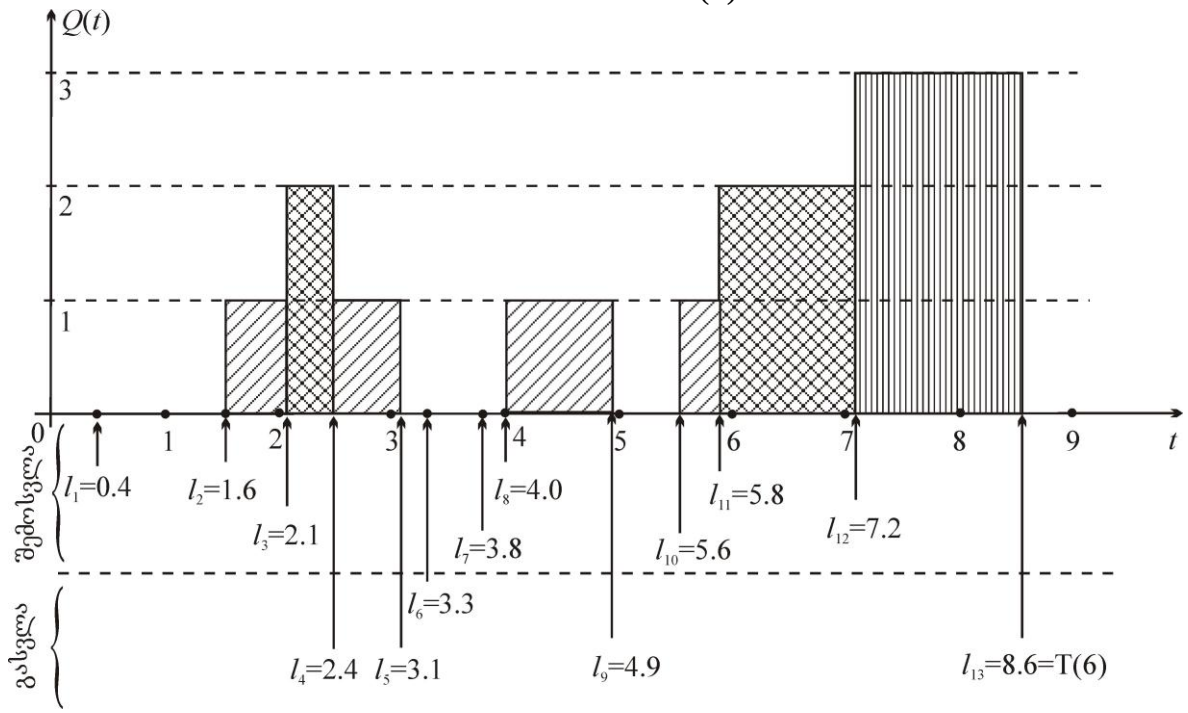
სადაც $\hat{p}_i$ დისკრეტული დროის ნაწილია, რომელიც იზომება *მოდელირების* პროცესში, როდესაც რიგში დაკავებულ მოთხოვნათა მნიშვნელობა *უტოლდებოდა* i-ს. მაგრამ $\hat{q}(n)$-ის გამოთვლისთვის მოსახერხებელია მოვახდინოთ (4.3)-ის ისეთი გარდაქმნა, რომელიც გამოიყენებს *უსასრულო ჯამის გეომეტრიულ ინტერპრეტაციას.* დავუშვათ, რომ T_i დროის ჯამური რიცხვია, რომელშიც მოდელირებისას რიგში დაკავებულ შეკვეთათა რაოდენობა რომ უტოლდებოდა i-ს. მაშინ

$$T(n) = T_0 + T_1 + T_2 + \cdots + , \tag{4.4}$$

$$\hat{p}_i = \frac{T_i}{T(n)} \,. \tag{4.5}$$

ფორმულა (4.3) შეიძლება ასე ჩავწეროთ:

$$\hat{q}(n) = \frac{\sum_{i=0}^{\infty} i T_i}{T(n)} \,. \tag{4.6}$$



ნახ. 4.2. $Q(t)$; მმსმეკ-ის რეალიზაციაში მოთხოვნის შემოსვლისა და გასვლის დროები

ნახ. 4.2-ზე ნაჩვენებია დროის შესაძლო მიმდინარეობა, ანუ $Q(t)$-ს რეალიზაცია სისტემაში, როცა $n=6$. დავუშვათ, რომ მოთხოვნები დგებოდა 0.4; 1.6; 2.1; 3.8; 4.0; 5.6; 5.8; 7.2 დროის მომენტებში, ხოლო მომსახურეობის დასრულება (სისტემიდან გასვლა) ხდებოდა დროის 2.4; 3.1; 3.3; 4.9; 8.6 მომენტებში; დროის $T(6)=8.6$ მომენტში მოდელირება შეწყდა.

ნახ. 4.2-დან $Q(t)$ არ ითვალისწინებს მოთხოვნებს, რომლებიც იმყოფებიან მომსახურეობაში (თუ ასეთი არსებობს). 0,4-დან 1,6 დროის ინტერვალში რიგი არ იქნება (თუ $t\in[0{,}4;1{,}6)$, მაშინ $Q(t)=0$), თუმცა ამ პერიოდში ერთი მოთხოვნა მომსახურეობაზეა, იგივე ხდება (3,1;3,3), (3,8;4,0) და (4,9;5,6) დროის ინტერვალებში, მაგრამ 3,1-დან 3,3 ინტერვალში მოთხოვნა საერთოდ არ არსებობს და სისტემა იმყოფება არადაკავებულ მდგომარეობაში, როგორც ეს იყო $[0;0{,}4]$ პერიოდში.

რომ გამოვთვალოთ $\hat{q}(n)$, თავიდან აუცილებელია T_i მნიშვნელობათა გამოთვლა. ნახ. 4.2-ის დახმარებით ის შეიძლება წარმოვიდგინოთ, როგორც დროის ინტერვალი, რომლის განმავლობაში $Q(t)=i$ ($i=0,1,2$ და ა.შ.):

$$
\begin{aligned}
T_0 &= (1{,}6-0{,}0)+(4{,}0-3{,}1)+(5{,}6-4{,}9)=3{,}2;\\
T_1 &= (2{,}1-1{,}6)+(3{,}1-2{,}4)+(4{,}9-4{,}0)+(5{,}8-5{,}6)=2{,}3;\\
T_2 &= (2{,}4-2{,}1)+(7{,}2-5{,}8)=1{,}7;\\
T_3 &= (8{,}6-7{,}2)=1{,}4;\\
T_i &= 0 \quad \text{თუ} \quad i\ge 4.
\end{aligned}
$$

ფორმულა (4.6)-ში მრიცხველი ასე გამოითვლება:

$$\sum_{i=0}^{\infty} iT_i = (0\times 3{,}2)+(1\times 2{,}3)+(2\times 1{,}7)+(3\times 1{,}4)=9{,}9. \qquad (4.7)$$

მოცემული მოდელის გატარებისას დროის ერთეულში რიგში მოთხოვნათა საშუალო მნიშვნელობის შეფასებაა:

$$\hat{q}(6)=9{,}9/8{,}6=1.115.$$

(4.7) ფორმულაში ყოველ არანულოვან შესაკრებს ნახ. 4.2-ზე შეესაბამება მართკუთხედები შესაბამისი ფართობებით: $1\times 2{,}3$-ს შეესაბამება ფართობი, რომელიც დიაგონალური ხაზებითაა დაშტრიხული, $2\times 1{,}7$-ს – ფართობი, რომელიც გადამკვეთი ხაზებითაა დაშტრიხული, $3\times 1{,}4$-ს – ფართობი, რომელიც შეფერადებულია მუქი ფერით. მოკლედ (4.7)-ის ჯამს შეესაბამება ნახ. 4.2-ზე $Q(t)$-ს ფუნქციის გრაფიკითა და $0x$ და $0y$ ღერძების შემოსაზღვრით მიღებული *მართკუთხედების ფართობების ჯამი*, რაც რიმანის განსაზღვრული ინტეგრალის განმარტებაა $Q(t)$-სთვის:

$$\sum_{i=0}^{\infty} iT_i \overset{\Delta}{=} \int_0^T Q(t)\,dt,$$

ხოლო $q(n)$-ის შეფასება კი ასე ჩაიწერება:

$$\hat{q}(n)=\frac{\int_0^T Q(t)\,dt}{T(n)}. \qquad (4.8)$$

შევნიშნოთ, რომ (4.8) ფორმულით სარგებლობა უკეთესია, რამეთუ *მოდელირებისას დაგროვებული ფართობების ჯამის თვლა რეკურენტული გზით უფრო მარტივია, ვიდრე* (4.5) *ფორმულით სარგებლობა,* სადაც *პარალელურად საჭიროა* T_i სიდიდეების თვლა.

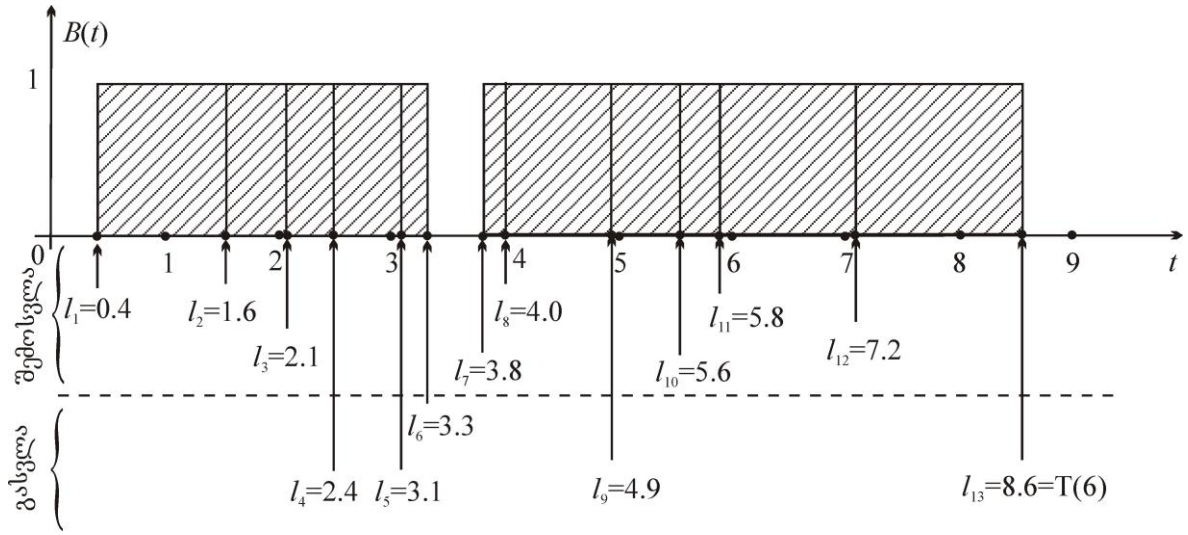
მესამე გამოსავალი კრიტერიუმია *კვანძის დაკავებულობის მახასიათებელი: კვანძით მოსალოდნელი სარგებლობის კოეფიციენტი* არის შეფარდება, რომლის მრიცხველია დრო,

რომლის განმავლობაშიც კვანძი იყო დაკავებული მომსახურეობაზე, ხოლო მნიშვნელია მოდელირების დრო – $T(n)$. მაშასადამე ეს კოეფიციენტი მოთავსებულია 0-სა და 1-ს შორის. აღინიშნება იგი $u(n)$-ით. სიმულაციის ერთი გატარებისთვის ითვლიან $u(n)$-ის შეფასებას – $\hat{u}(n)$, რომელიც წარმოადგენს $u(n)$-ის განმარტების დროთა შერჩევით შეფასებათა შეფარდებას. $\hat{u}(n)$-ის უკეთ განმარტებისთვის შემოვიღოთ შემდეგი ფუნქცია: $B(t)$ – *კვანძის დაკავების ფუნქცია*:

$$B(t)=\begin{cases}1, & \text{თუ კვანძი დაკავებულია } t \text{ მომენტში},\\ 0, & \text{თუ კვანძი თავისუფალია } t \text{ მომენტში}.\end{cases}$$

ე.ი. $\hat{u}(n)$ კოეფიციენტი გამოხატავს იმ დროის ნაწილს, როცა $B(t)\equiv 1$. ნახ. 4.3-ზე გამოსახულია სისტემის მოდელირების გასწვრივ $B(t)$-ს გრაფიკი, რომელიც ნახ. 4.2-დან მიიღება:

$$\hat{u}(n)=\frac{(3{,}3-0{,}4)+(8{,}6-3{,}8)}{8{,}6}=\frac{7{,}7}{8{,}6}=0{,}9. \qquad (4.9)$$



ნახ. 4.3. $B(t)$; *მმსმეკ-ის რეალიზაციაში მოთხოვნის შემოსვლისა და გასვლის დროები*

გამოთვლების შედეგი მიუთითებს, რომ მოდელირების დროის 90%-ში კვანძი დაკავებულია მომსახურეობაზე. მაგრამ მრიცხველი (4.9)-ში ასევე შეიძლება შეიცვალოს ფართობების ჯამით, ანუ $B(t)$-დან განსაზღვრული ინტეგრალით:

$$\hat{u}(n)=\frac{\int_0^{T(n)} B(t)\,dt}{T(n)}. \qquad (4.10)$$

აქაც $\hat{u}(n)$-ის გამოთვლა (იგივე მიზეზით, რაც ადრე $\hat{q}(n)$-სთვის) ინტეგრალური ფორმულით უფრო მისაღებია სიმულაციის პროცესში.

მოდელირების მრავალ შემთხვევაში, როდესაც სისტემაში გამოიყენება სხვადასხვა ტიპის მომსახურეობის კვანძები, კვანძით სარგებლობის შეფასების კოეფიციენტი შეიცავს საკმარის სტატისტიკას, რათა გადაწყდეს პრობლემა რესურსების დეფიციტისა. თუ კოეფიციენტი $\approx 100\%$-ია, ეს მიუთითებს კვანძის მაღალ დაკავებულობას და მოთხოვნათა გრძელ რიგს. ეს განსაკუთრებით ეხება შემთხვევებს, როდესაც მომსახურეობის კვანძებია ისეთი ძვირადღირებული მოწყობილობანი, როგორიცაა საწარმოო სისტემების რობოტები ან დიდი კომპიუტერები (მეინფრეიმები), რომლებიც კავდებიან მონაცემთა დამუშავების ოპერაციებში.

მაშასადამე სისტემის მუშაობის შეფასების კრიტერიუმებად განვიხილეთ:

1) $\hat{d}(n)$ – რიგში მოთხოვნის დაკავების საშუალო დრო;
2) $\hat{q}(n)$ – რიგში მოთხოვნათა საშუალო სიდიდე;
3) $\hat{u}(n)$ – კვანძით სარგებლობის კოეფიციენტი.

მოთხოვნის დაკავების საშუალო დრო დისკრეტული ტიპის სტატისტიკაა; რიგში მოთხოვნათა საშუალო სიდიდე უწყვეტი ტიპის სტატისტიკაა; კვანძით სარგებლობის კოეფიციენტი ასევე უწყვეტი ტიპის სტატისტიკაა. უკანასკნელი ორის განსაზღვრებაში მონაწილეობს $Q(t)$, $B(t)$ ფუნქციები. გარდა ამ სტატისტიკებისა, მასობრივი მომსახურეობის სისტემების მუშაობის შეფასების კრიტერიუმად გამოიყენება სხვა სტატისტიკებიც, როგორიცაა: მაქსიმალური დაგვიანება რიგში; მოდელირების დროის წილი, როცა რიგში იმყოფებოდა არანაკლებ k მოთხოვნა $(0 < k < n)$ და ა.შ.

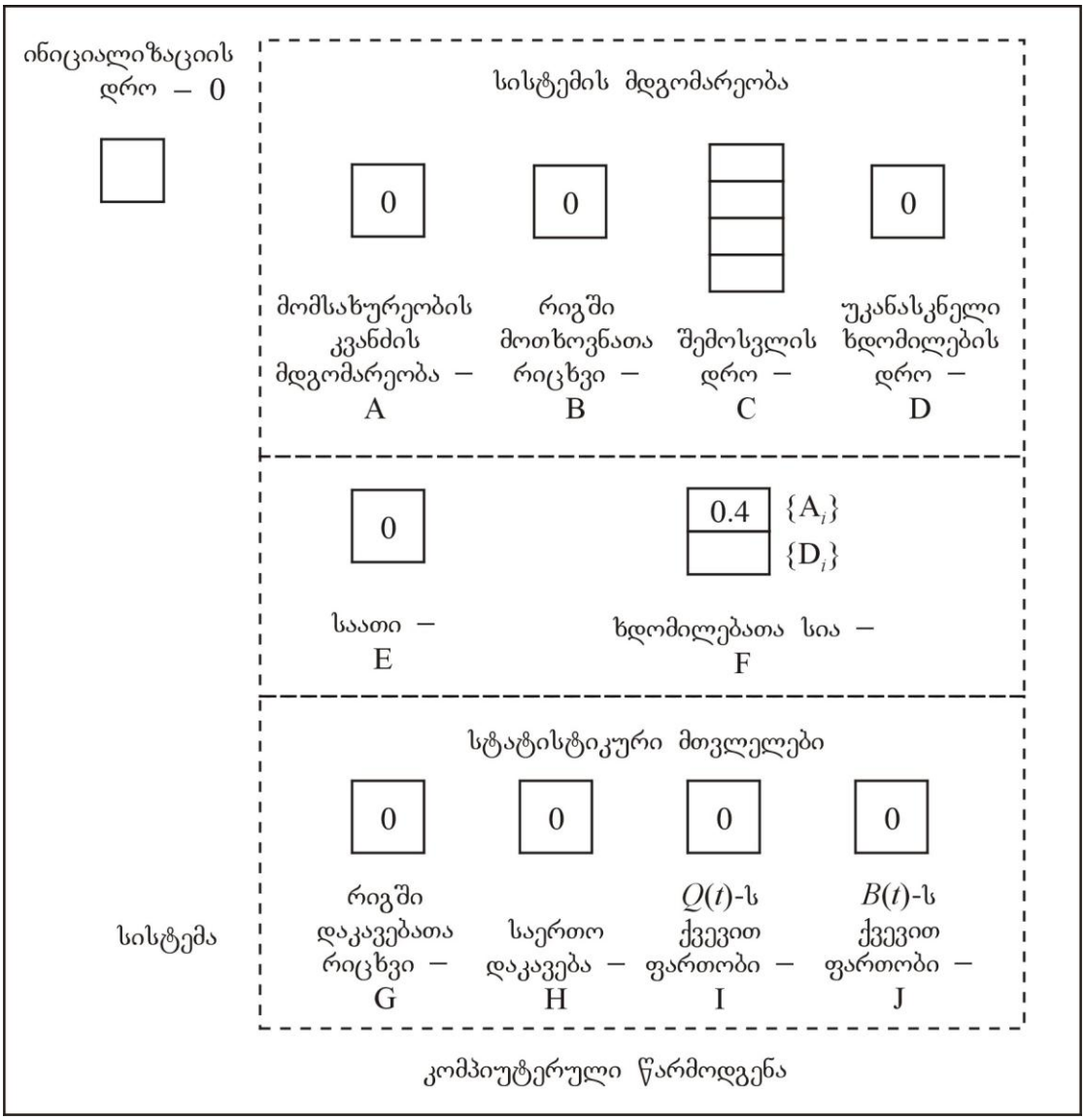
და ბოლოს, მმსმეკ-ის ხდომილობებია: მოთხოვნის შემოსვლა და გასვლა. $d(n)$, $q(n)$ და $u(n)$ კრიტერიუმთა შეფასებისთვის აუცილებელი ცვლადებია: კვანძის მომსახურეობის მდგომარეობა (დაკავებულია – თავისუფალია); რიგში მოთხოვნათა რიცხვი; ყოველი მოთხოვნის დადგომის მომენტი, რომელიც მიმდინარე მომენტში რიგშია (ჩაწერილია სიაში); უკანასკნელი ხდომილების დადგომის დრო.

§4.2. მმსმეკ-ის სიმულაციური მოდელირების ინტუიციური წარმოდგენა

მმსმეკ-ს მოდელირების პროცესის ახსნისთვის დასაწყისში განვიხილოთ დროის კონკრეტული მიმდინარეობა. დავუშვათ სიმულაციური მოდელი კომპიუტერში წარმოიდგინება $e_0 = 0, e_1, e_2, \ldots, e_{13}$ დროის მომენტებში, როდესაც დგება ცამეტი მიმდევრობითი ხდომილება. რიგში აუცილებელია $n = 6$ მოთხოვნათა დაკავების დაფიქსირება მოდელირების პროცესის გაჩერებისთვის. მოხერხებულობის მიზნით მივიღოთ, რომ მოთხოვნების დადგომების შორის დრო და მოთხოვნაზე მომსახურეობის დროის მიმდევრობაა:

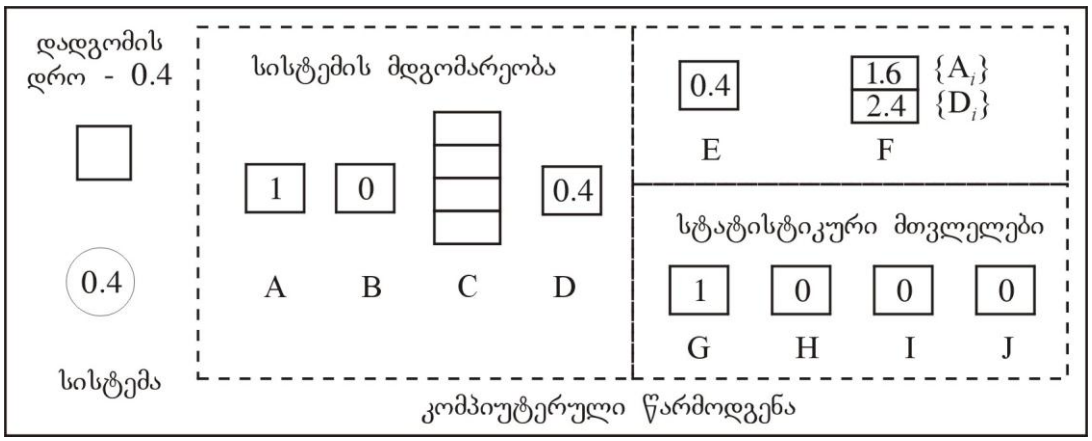
$$A_1 = 0.4,\ A_2 = 1.2,\ A_3 = 0.5,\ A_4 = 1.7,\ A_5 = 0.2,$$
$$A_6 = 1.6,\ A_7 = 0.2,\ A_8 = 1.4,\ A_9 = 1.9,\ \ldots$$
$$S_1 = 2.0,\ S_2 = 0.7,\ S_3 = 0.2,\ S_4 = 1.1,\ S_5 = 3.7,\ S_6 = 0.7,\ \ldots$$

როგორც უკვე აღვნიშნეთ, $\{A_i\}$ და $\{S_i\}$ მნიშვნელობები გენერირდებიან შესაბამისი ალბათური განაწილებიდან. აქ მოყვანილი მათი მნიშვნელობები ისეა შერჩეული, რომ ჩატარდეს ნახ. 4.2 და ნახ. 4.3-ზე დროის მიმდინარების შესაბამისი მმსმეკ, რომლის $Q(t)$ და $B(t)$ ფუნქციები განისაზღვრებიან სისტემის სიმულაციის პროცესში. ნახ. 4.4-ზე ნაჩვენებია სისტემის სქემები და მისი კომპიუტერული წარმოდგენა ყოველ e_i $(i = 0,1,\ldots,13)$ მომენტებში. სისტემის სქემებში კვადრატი აღნიშნავს მომსახურეობის კვანძს, ხოლო წრე – მოთხოვნას, წრეში ჩაწერილი რიცხვი – მოთხოვნის დადგომის დროს. სისტემის წარმოდგენის კომპიუტერულ სქემებზე ნაჩვენებია ცვლადების მნიშვნელობები, რომლებიც მიღებულია ხდომილებათა მონაცემების დამუშავების დასრულების შემდეგ. განვიხილოთ, თუ როგორ იცვლება კომპიუტერული წარმოდგენა ხდომილებათა წარმოქმნის (დადგომის) მომენტებში:

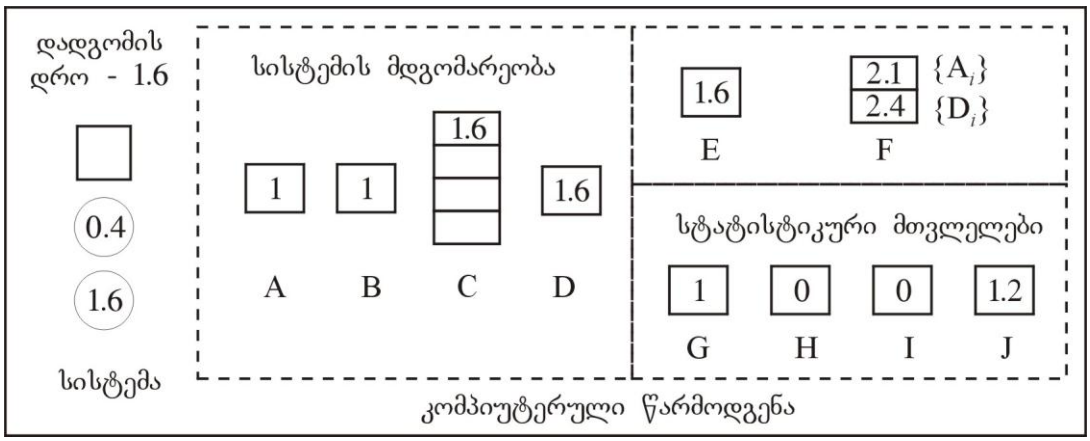


ნახ. 4.4.ა – ინიციალიზაცია

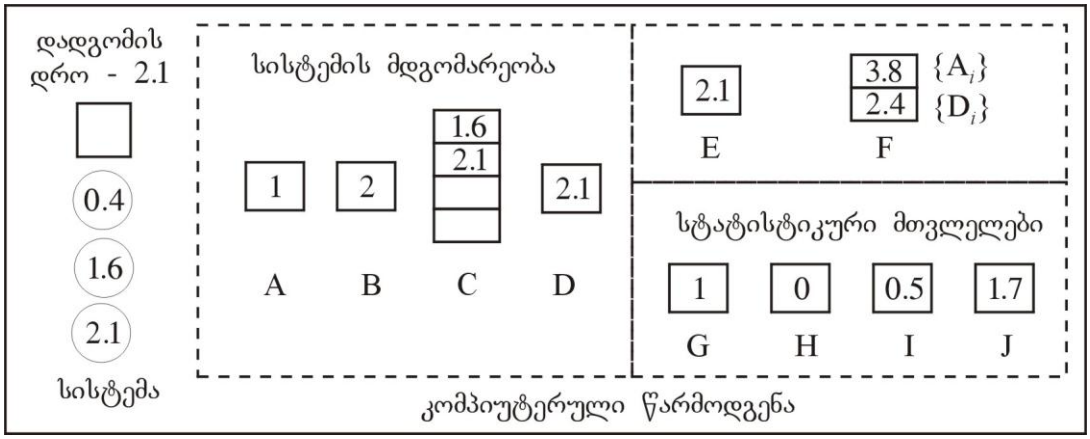
შენიშვნა. ნახ. 4.4-ის შემდეგ წარმოდგენებში გამოყენებულია გამონათქვამთა შესაბამისი საიდენტიფიკაციო ცვლადები $(A, B, C, D, E, F, G, H, I, J)$.



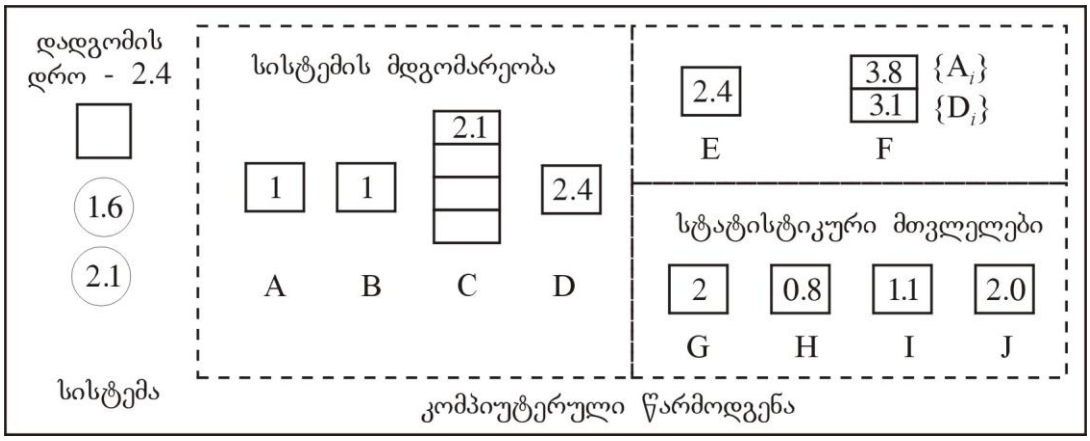
ნახ. 4.4.ბ – I ხდომილება. I ხდომილების დადგომა



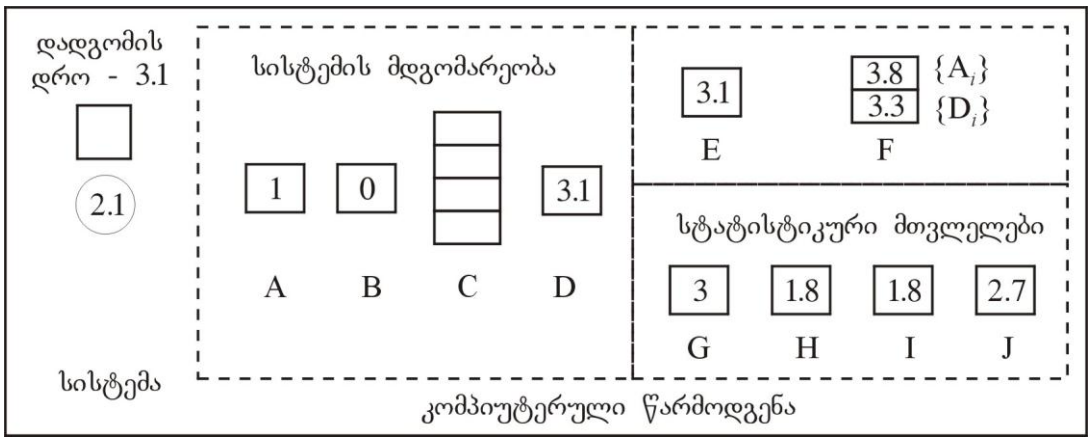
ნახ. 4.4.გ – II ხდომილება. II ხდომილების დადგომა



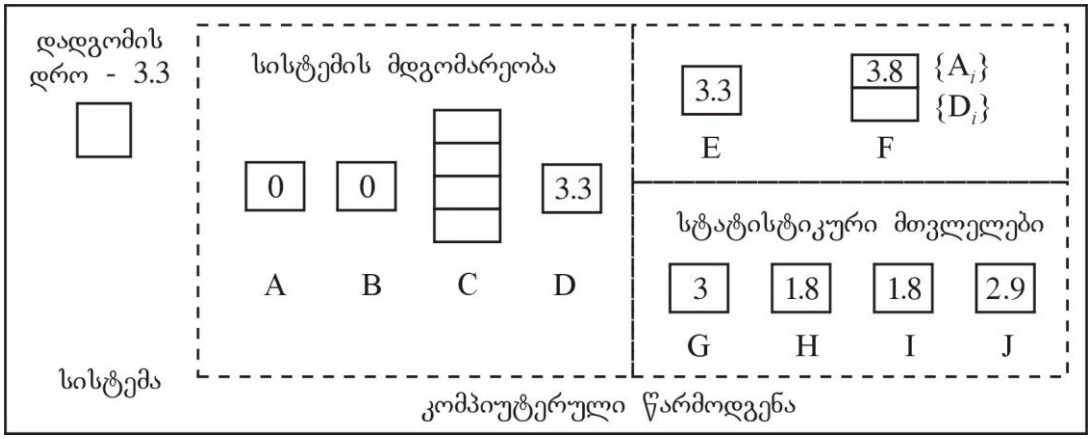
ნახ. 4.4.დ – III ხდომილება. III ხდომილების დადგომა



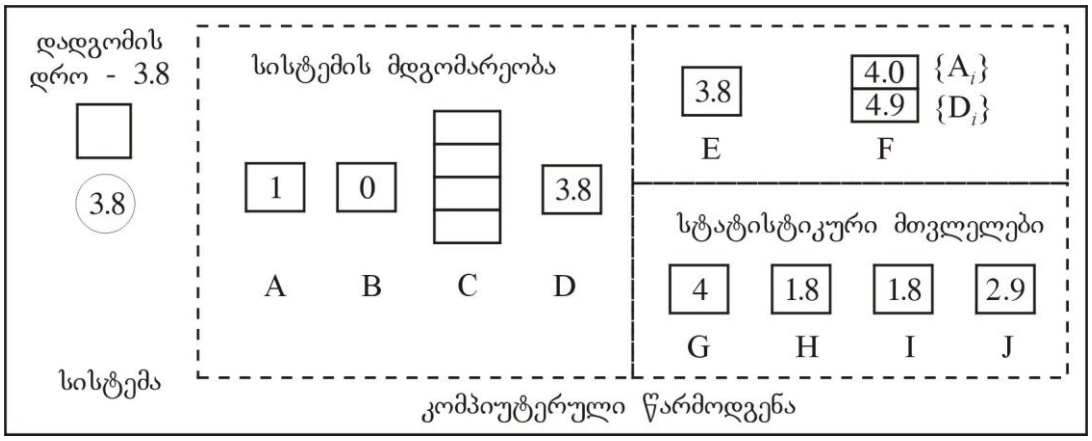
ნახ. 4.4.ე – IV ხდომილება. I მოთხოვნის გასვლა



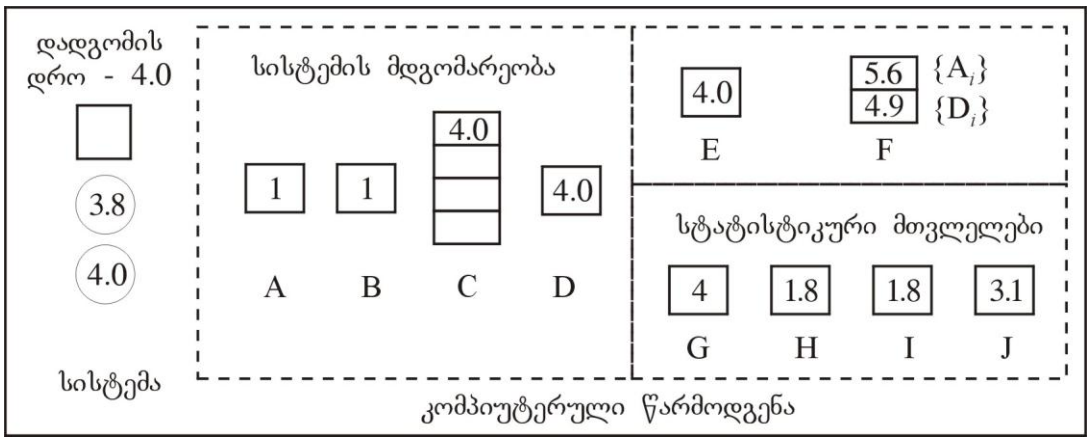
ნახ. 4.4.ვ – V ხდომილება. II მოთხოვნის გასვლა



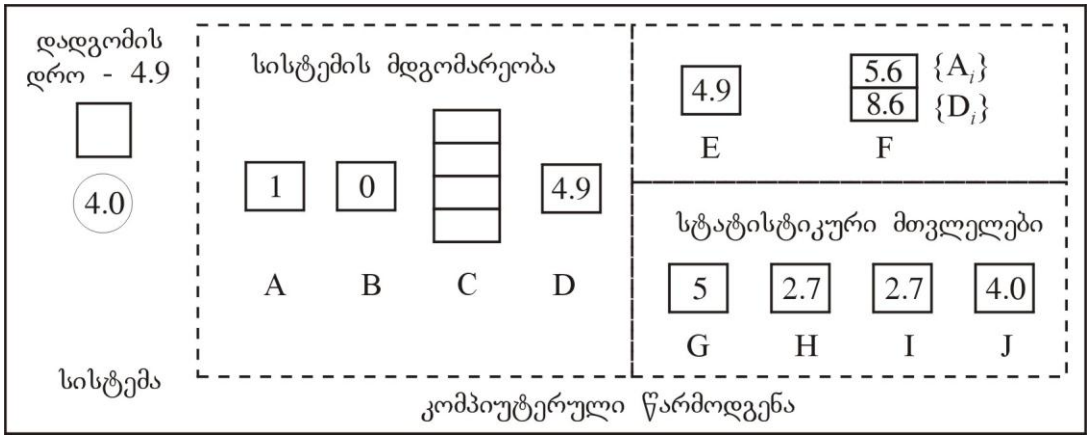
ნახ. 4.4.ზ – VI ხდომილება. III მოთხოვნის გასვლა



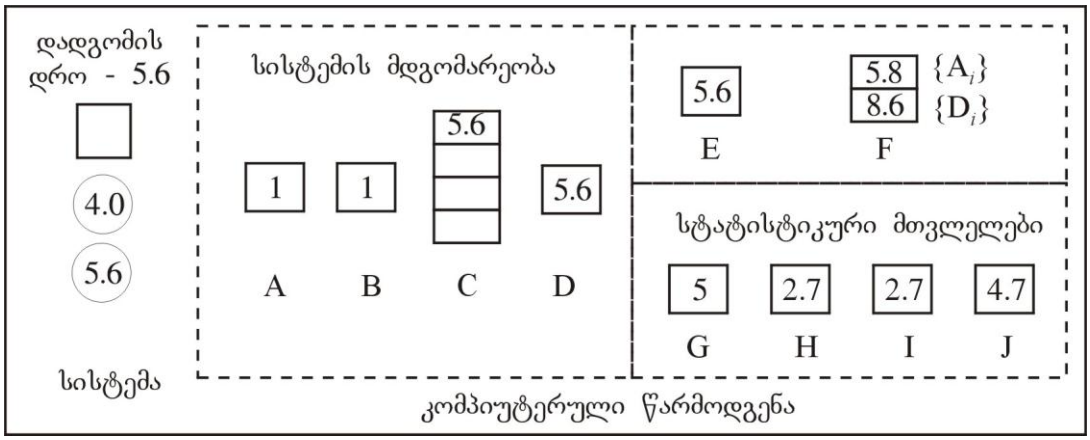
ნახ. 4.4.თ – VII ხდომილება. IV მოთხოვნის დადგომა



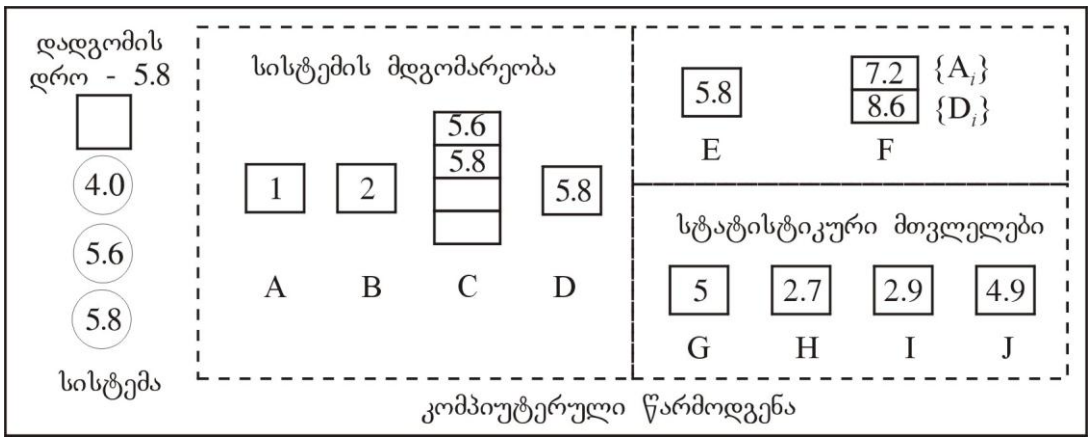
ნახ. 4.4.ი – VIII ხდომილება



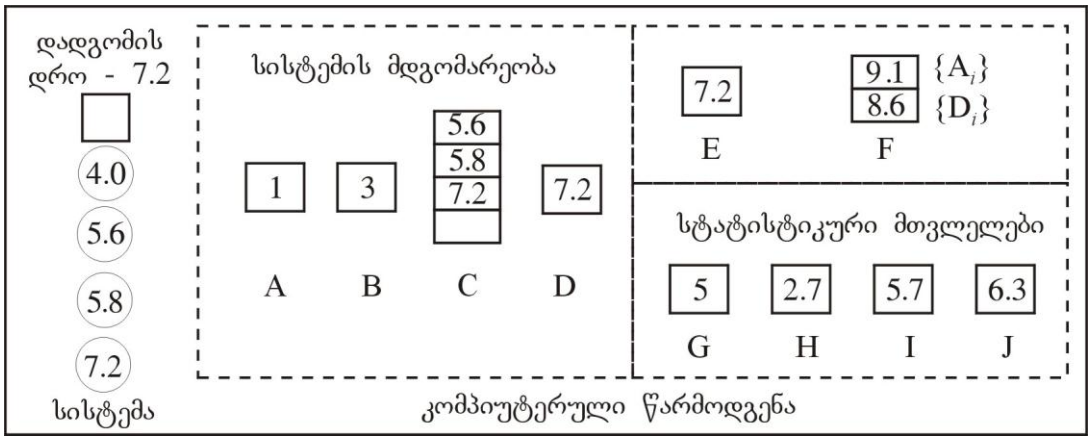
ნახ. 4.4.კ – IX ხდომილება



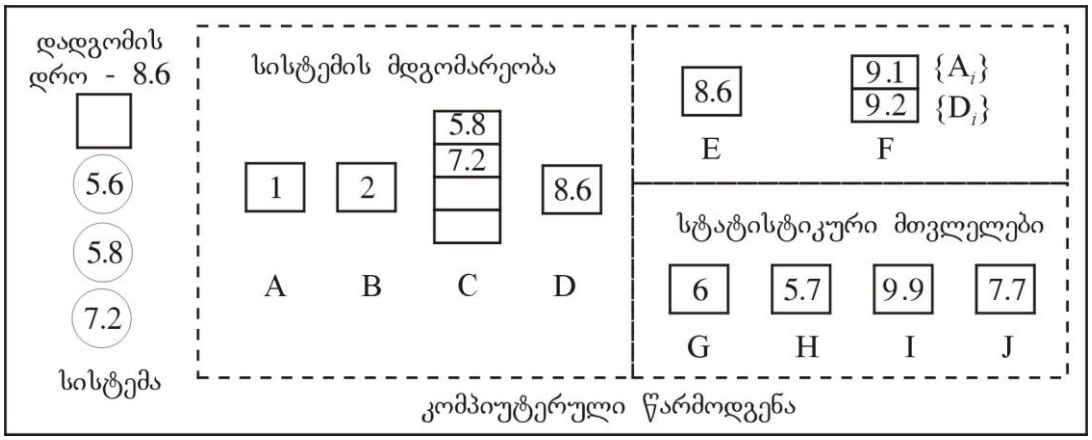
ნახ. 4.4.ლ – X ხდომილება



ნახ. 4.4.მ – XI ხდომილება



ნახ. 4.4.ნ – XII ხდომილება



ნახ. 4.4.ო – XIII ხდომილება

ინიციალიზაცია $(t = 0)$. მოდელირება იწყება ძირითადი პროგრამის მიერ ინიციალიზაციის პროგრამის გამოძახებით. იწყება რა მოდელირება, დაშვებულია, რომ სისტემაში დასაწყისისთვის არ იმყოფება არც ერთი მოთხოვნა და მომსახურეობის კვანძი თავისუფალია, როგორც ეს სქემის 4.4.ა ნაწილშია. მოდელის მდგომარეობის ცვლადები დაყენებულია საწყის პოზიციაში; კვანძის მდგომარეობა მომსახურეობაზე ტოლია 0-ის და რიგში მოთხოვნათა რიცხვიც ტოლია 0-ის. სისტემაში შემოსული მოთხოვნის დადგომის დროის მნიშვნელობის შესანახად, რომელიც *მოცემულ მომენტში რიგში იმყოფება*, გამოიყენება *ერთგანზომილებიანი მასივი*. ეს მასივი თავიდან ცარიელია, მაგრამ მოდელირების პროცესში მისი ზომა ხან იზრდება, ხან მცირდება (ბლოკი C). უკანასკნელი ხდომილების დადგომის

დრო ყენდება 0–ვან მდგომარეობაში (ბლოკი D). მოდელური დროის საათიც დაყენებულია 0-ვან მდგომარეობაში (ბლოკი E). ხდომილებათა სია, რომელიც ასახავს ყოველი ტიპის ხდომილების შემდეგი წარმოქმნის დროს, ინიციალიზირდება ასე: პირველი მოთხოვნის დადგომის დრო: $0 + A_1 = 0.4$ (ბლოკი F-ის ზედა ნაწილი). რადგან ამ მომენტში მოთხოვნა მომსახურეობაზე არ არის, აზრი არ აქვს ვილაპარაკოთ შემდეგი მოთხოვნის გასვლაზე (ბლოკი F, ქვედა ნაწილი). მოდელირების გაგრძელებისთვის ხდომილებათა სიიდან ირჩევა უმცირესი მნიშვნელობა, რომელიც განსაზღვრავს შემდეგი ხდომილების დადგომას. ამის შემდეგ ოთხი სტატისტიკური მთვლელი დაყენდება 0-ვან მდგომარეობაში. ინიციალიზაციის დასასრულს მართვა უბრუნდება ძირითად პროგრამას, რომელიც უკვე იძახებს სინქრონიზაციის პროგრამას, რათა მან განსაზღვროს შემდეგი ხდომილების დადგომა. შემდეგი ხდომილება იქნება მოთხოვნის დადგომა 0,4 მომენტში, ხოლო სინქრონიზაციის პროგრამას მდს გადაყავს ამ დროზე. მართვა კვლავ უბრუნდება ძირითად პროგრამას იმ ინფორმაციით, რომ შემდეგი ხდომილება კვლავ მოთხოვნის დადგომა იქნება.

მოთხოვნის დადგომა ($t = 0{,}4$). დროის 0,4 მომენტში ძირითადი პროგრამა მართვას გადასცემს ხდომილების დამუშავების პროგრამას. ის იწყებს პირველი მოთხოვნის დადგომის დამუშავებას. ნახ. 4.4ბ-ზე ნაჩვენებია სისტემა და მისი კომპიუტერული წარმოდგენა *ყველა ცვლილების შეტანის შემდეგ*, რომელიც გულისხმობს დადგომის დამუშავებას. რადგან ამ ხდომილების დადგომის მომენტში კვანძი იმყოფებოდა თავისუფალ მდგომარეობაში (0-ვანი მნიშვნელობა), მისი მომსახურეობა იწყება მყისიერად და რიგში დაკავება შეადგენს $D_1 = 0$ (თუმცა ასეთი დრო სინამდვილეში განიხილება როგორც დაკავება!). კვანძის მომსახურეობის მდგომარეობა ყენდება 1 მნიშვნელობაზე, რაც კვანძის დაკავებას ნიშნავს, თუმცა რიგი კვლავ ცარიელია. როგორც აღვნიშნეთ, მდს დაყენებულია 0,4-ზე და ხდომილებათა სია განახლდება ამ მოთხოვნის დადგომის გათვალისწინებით. შემდეგი დადგომა მოხდება $A_2 = 1{,}2$ დროის შემდეგ მიმდინარე 0,4 მომენტიდან, $0{,}4 + 1{,}2 = 1{,}6$ დროის მომენტში, ხოლო მორიგი შემდეგი გასვლა სისტემიდან მოხდება $S_1 = 2{,}0$ დროის შემდეგ მიმდინარე 0,4 მომენტიდან, $0{,}4 + 2 = 2{,}4$ დროის მომენტში. რიგში დაკავებულ მოთხოვნათა რიცხვი 1-ით იზრდება (რადგან $1{,}6 < 2{,}4$ და მეორე შემოსული მოთხოვნა რიგში აღმოჩნდება). როდესაც რიგში მოთხოვნათა საერთო რიცხვი მიაღწევს $n = 6$-ს, მაშინ მოდელირება შეწყდება. და $D_1 = 0$ დაემატება საერთო დაგვიანების დროს, რომელიც ჯერ კიდევ 0-ის ტოლია. ნახ. 4.2 და ნახ. 4.3-ის კვლავ განხილვით $Q(t)$ და $B(t)$ მნიშვნელობები მდს-ს მიმდინარე ჩვენებისას 0-ის ტოლია. აქ უკანასკნელი ხდომილების დრო (ბლოკი D) უტოლდება მდს-ის ჩვენებას – 0,4 და მართვა კვლავ ძირითად პროგრამას გადაეცემა. ის იძახებს სინქრონიზაციის პროგრამას, რომელიც დაათვალიერებს ხდომილებათა სიას და უმცირესი მნიშვნელობიდან განსაზღვრავს, თუ რომელი ხდომილება დადგება შემდეგში. ეს დრო კი 1.6-ის ტოლია, რომელიც მეორე მოთხოვნის შემოსვლას ნიშნავს. პროგრამა განაახლებს მდს-ს მნიშვნელობას (მდს $= 1{,}6$) და ამ მნიშვნელობით მართვას გადასცემს ძირითად პროგრამას იმ ინფორმაციით, რომ შემდეგი ხდომილება იქნება მოთხოვნის დადგომა.

მეორე მოთხოვნის დადგომა ($t = 1{,}6$). კვლავ გამოიძახება ხდომილებათა დამუშავების პროგრამა. ნახ. 4.4გ-ზე ნაჩვენებია სისტემა და მისი კომპიუტერული წარმოდგენა ყველა ცვლილების შეტანის შემდეგ, რომელიც გულისხმობს ხდომილების დადგომის დამუშავებას. რადგან ამ მოთხოვნის დადგომისას მომსახურეობის კვანძი იმყოფება დაკავებულ მდგომარეობაში (მდგომარეობა $= 1$), ის რიგში პირველი უნდა ჩადგეს. მისი შემოსვლის დრო ინახება მასივის პირველ უჯრედში და რიგში დაკავებულ მოთხოვნათა რიცხვის ცვლადი 1-ით იზრდება. ხდომილებათა სიაში შემდეგი ხდომილების დადგომის დრო იზრდება $A_3 = 0{,}5$ დროით მიმდინარე მომენტიდან (მდს $= 1{,}6$) და უტოლდება $1{,}6 + 0{,}5 = 2{,}1$-ს. შემდეგი გასვლის დრო არ იცვლება, რადგან ის შეადგენს 2,4-ს და იგი პირველი მოთხოვნის გასვლის დროა, რომელიც ჯერ კიდევ მომსახურეობაშია (2,4 > მდს). რადგან პროგრამა *არ ელოდება*

რიგში რომელიმე დაკავების დასრულებას, ამიტომ რიგში დაკავებათა რიცხვი და საერთო დაკავების დრო არ იცვლება. $Q(t)$-თი შემოსაზღვრული ფართობი იზრდება 0-ით, ფართობი, შემოსაზღვრული $B(t)$-თი, იზრდება 1-ით. ბოლო ხდომილების დადგომის დროის განახლების შემდეგ, მართვა კვლავ გადაეცემა ძირითად პროგრამას. ის თავის მხრივ მართვას გადასცემს სინქრონიზაციის პროგრამას, რომელიც განსაზღვრავს, რომ შემდეგი ხდომილება იქნება მოთხოვნის დადგომა $2{,}1$ მომენტში.

მესამე მოთხოვნის დადგომა $(t = 2{,}1)$. კვლავ გამოიძახება ხდომილების დამუშავების პროგრამა (ნახ. 4.4.დ). რადგან $2{,}4 > 2{,}1$, მომსახურეობის კვანძი დაკავებულია და რიგი იზრდება კიდევ ერთი მოთხოვნით. მისი დადგომის დრო შეინახება მართვის მეორე უჯრედში. შემდეგი დადგომის დრო განახლდება $t + A_4 = \text{მდს} + A_4 = 2{,}1 + 1{,}7 = 3{,}8$-ით. ხოლო შემდეგი გასვლის დრო ჯერ კიდევ უცვლელია, რადგან ველოდებით პირველ მოთხოვნაზე მომსახურეობის დასრულებას. დაკავების მთვლელების მნიშვნელობები უცვლელია, რადგან არც ერთი დაკავება რიგში არ დამთავრებულა, ხოლო ფართობების მთვლელების მნიშვნელობებს ერთი მიემატებათ. უკანასკნელი ხდომილების დადგომის დროის მაჩვენებლის მიმდინარეთი შეცვლის შემდეგ მართვა უბრუნდება ძირითად პროგრამას, ის კი იძახებს სინქრონიზაციის პროგრამას, რომელიც დაათვალიერებს ხდომილებათა სიას და განსაზღვრავს, რომ შემდეგი ხდომილება იქნება მოთხოვნის გასვლა $2{,}4$ დროის მომენტში და მდს-ს განაახლებს ამ დროზე.

პირველი მოთხოვნის გასვლა $(t = 2{,}4)$. გამოიძახება ხდომილებათა დამუშავების პროგრამა და მისი ნაწილი – მოთხოვნის გასვლა. კვანძი რჩება დაკავების მდგომარეობაში, რადგან მოთხოვნა 2 რიგში პირველი ადგილიდან გადადის მომსახურეობაზე. რიგი ერთით მცირდება, ხოლო ხდომილებათა დადგომის დროის მასივი გადაადგილდება ერთი უჯრით და ამით აჩვენებს, რომ მოთხოვნა 3 ახლა პირველია რიგში. მოთხოვნა 2, რომელიც გადადის მომსახურეობაზე $S_2 = 0{,}7$ დროით და შემდეგი გასვლის დრო ხდომილებათა სიაში იზრდება S_2-ით მიმდინარე დროიდან და ის $2{,}4 + 0{,}7 = 3{,}1$-ს უტოლდება. დაკავების სტატისტიკა განახლდება, რადგან ამ მომენტისთვის მოთხოვნა 2 გადადის მომსახურეობაზე და ასრულებს ლოდინს რიგში. ამ შემთხვევაში ვისარგებლებთ დადგომის დროის მასივით და დავითვლით მოთხოვნა 2-ის რიგში დაკავების ხანგრძლივობას: $D_1 = 2{,}4 - 1{,}6 = 0{,}8$ (1,6 მნიშვნელობა, მე-2 მოთხოვნის შემოსვლის მომენტი, ინახებოდა მასივის I უჯრედში, ვიდრე მოხდებოდა მასივის განახლება, ამიტომ ეს გამოთვლა უნდა შესრულდეს მასივის განახლებამდე). $Q(t)$ და $B(t)$ ფუნქციებით შემოსაზღვრული ფართობების განახლება ასეთია: $Q(t)$-ს დაემატება $2 \times (2{,}4 - 2{,}1)$, ხოლო $B(t)$-ს დაემატება $1 \times (2{,}4 - 2{,}1)$. უკანასკნელი ხდომილების დრო განახლდება და მართვა კვლავ უბრუნდება ძირითად პროგრამას. შემდეგ მართვა გადაეცემა სინქრონიზაციის პროგრამას, რომელიც განსაზღვრავს, რომ შემდეგი ხდომილება იქნება მოთხოვნის გასვლა $t = 3{,}1$ მომენტში.

მეორე მოთხოვნის გასვლა $(t = 3{,}1)$. ცვლილებები ანალოგიურია პირველი მოთხოვნის გასვლისა $t = 2{,}4$ დროის მომენტში. მივაქციოთ ყურადღება იმას, რომ ვაკვირდებით კიდევ ერთ დაკავებას რიგში და ამ ხდომილების შემდეგ რიგი კვლავ ცარიელი ხდება, ხოლო კვანძი კი დაკავებული.

მესამე მოთხოვნის გასვლა $(t = 3{,}3)$. ცვლილებები ანალოგიურია, ვიდრე პირველ და მეორე მოთხოვნების გასვლებისა. თუმცა ამ შემთხვევაში იქნება არცთუ უმნიშვნელო გამონაკლისი – რადგან რიგი ცარიელია, ამიტომ კვანძი (მოწყობილობა) ხდომილების შემდეგ თავისუფალია. სისტემა ისევე გამოიყურება, როგორც 0-ვან მომენტში, ხოლო პროგრამა გადადის იქითკენ, რომ შემდეგი ხდომილება იქნება მე-4 მოთხოვნის დადგომა.

მეოთხე მოთხოვნის დადგომა $(t = 3{,}8)$. რადგან ეს მოთხოვნა აფიქსირებს კვანძს, როგორც თავისუფალს, მისი დაკავება რიგში 0-ის ტოლია $(D_4 = 0)$. ამიტომ ცვლილებები ამ შემთხვევაში ანალოგიურია იმისა, რაც იყო პირველი მოთხოვნის დადგომისას $t = 0{,}4$ მომენტში. დარჩენილი 6 ხდომილება, რომელიც გამოსახულია ნახ. 4.4ი-დან დამთავრებული

ნახ. 4.4ო-მდე, საჭიროა ანალოგიურად განიხილოთ და თვითონ დაასრულოთ ეს დიალოგი-სქემა. დარწმუნდით, რომ ჩაწერილი შესაბამისი სიდიდეები შეესაბამება რეალობას. მოდელირების გატარებაში დაგეხმარებათ ნახ. 4.2 და ნახ. 4.3-ზე წარმოდგენილი $Q(t)$ და $B(t)$ ფუნქციათა გრაფიკები. მეხუთე მოთხოვნის გასვლით დროის $t = 8{,}6$ მომენტში მოთხოვნა 6 ტოვებს რიგს და გადადის მომსახურეობაში. მაშასადამე დაკავებულ მოთხოვნათა რიცხვი რიგში აღწევს 6-ს (n -ის წინასწარი მნიშვნელობა) და ამით მოდელირება მთავრდება. ამ მომენტში ძირითადი პროგრამა *იძახებს ანგარიშების გენერატორს*, რათა მან გამოთვალოს სისტემის მუშაობის შეფასების გამოსავალი მახასიათებლები: $\hat{d}(6) = 5{,}7/6 = 0.95$; $\hat{q}(6) = 9{,}9/8{,}6 = 1.15$; $\hat{u}(6) = 7{,}7/8{,}6 = 0.90$.

მაგალითი, რომელმაც წარმოგვიდგინა მოდელირების ლოგიკა, ითხოვს კომენტარებს:

- ალბათ, მოდელირების დინამიკის ერთ-ერთ მთავარ საკვანძო ელემენტს წარმოადგენს მდს-სა და ხდომილებათა სიის ურთიერთქმედება. საათი გადაიყვანება შემდეგი ხდომილების დროზე, რომელიც განისაზღვრება ყოველი ხდომილების დამუშავების ბოლოში ხდომილებათა სიის სკანირებითა და მასში უმცირესი დროის მქონე ხდომილების პოვნით. ასე ხორციელდება მოდელირების მიმდინარეობა დროში.
- ხშირად წარმოიქმნება გაუთვალისწინებელი გარემოებები, რომლებიც პროგრამაში მაინც უნდა იყოს გათვალისწინებული. მაგალითად, არ დავივიწყოთ, რომ მოთხოვნის გასვლის შემდეგ რიგი შეიძლება ცარიელი შეიქნას. ეს კი გამოიწვევს კვანძის დაუკავებლობას მომსახურეობაზე. საიდანაც გამომდინარეობს, რომ *მოთხოვნის გასვლის ხდომილების აღრიცხვა აუცილებლად უნდა გადაიდოს.* არსებობს მოდელირების გაჩერების უფრო რთული პირობებიც, ვიდრე ეს ერთი შეხედვით შეიძლება გვეჩვენოს!
- ზოგიერთ სიმულაციურ მოდელში შეიძლება აღმოჩნდეს, რომ ორი (ან მეტი) ელემენტი ხდომილებათა სიიდან *დამოუკიდებლად და ერთდროულად* განსაზღვრავს შემდეგი ხდომილების დადგომის დროს, ანუ უმცირეს დროს (მაგალითად, საუბარი შეიძლება იყოს ერთდროულ ხდომილებებზე, რომლებიც წამოიქმნება ერთი და იგივე მოდელურ დროში). ეს შეიძლება იყოს ხდომილება, რომელიც აღწერს მოთხოვნაზე მომსახურეობის დასრულებას და ხდომილება, რომელიც აღწერს ახალი მოთხოვნის დადგომას). ასეთი *დროითი კავშირის* დარღვევის თავიდან ასაცილებლად ჩართული უნდა იყოს *გადამწყვეტი წესი*, რომელიც კონკრეტული ამოცანებიდან *მიღება უპირატესობის განმარტებით.* გადამწყვეტი წესი როგორც ყოველთვის ზემოქმედებს მოდელირების რეზულტატებზე.

ზემოთ მოყვანილი სავარჯიშო აკეთებს დემონსტრირებას, თუ როგორი ცვლილებები და მონაცემთა სტრუქტურები მოქმედებენ დისკრეტულ-ხდომილებით სიმულაციურ მოდელირებაში ხდომილებათა დაგეგმვის თვალსაზრისით. მასში წარმოდგენილია ყველაზე მნიშვნელოვანი იდეები, რომლებიც აუცილებელია გაცილებით რთული სისტემების მოდელირებისას. მოთხოვნებს შორის დრო და მოთხოვნის მომსახურეობაზე დახარჯული დრო მიღებულია შემთხვევითი რიცხვების ცხრილიდან. *"ხელით" ჩატარებული მოდელირება* შეიძლება აღმოჩნდეს რაგინდ გაგრძელებადი, რაც კრიტერიუმების შეფასებებს გაამაგრებდა, მაგრამ ის მალე აღმოჩნდება დამღლელი და ამიტომ მივმართავთ კომპიუტერს არითმეტიკული ოპერაციების ჩატარებისთვის და სისტემური რესურსების აღრიცხვისათვის. განსაკუთრებით მაშინ, როდესაც საჭიროა *რთული სისტემების სიმულაცია.*

§4.3. პროგრამის ორგანიზაცია და ლოგიკა

ამ პარაგრაფში განვსაზღვრავთ კომპონენტებს, რომლებიც აუცილებელია მმსმეკ-ის მოდელირებისთვის უნივერსალურ ენა C-ზე. პროგრამის ორგანიზაცია და ლოგიკა, რომელიც

აქ იქნება წარმოდგენილი, ეხება შემდეგ პარაგრაფში წარმოდგენილ პროგრამულ კოდს (ამარტივებს კოდის წაკითხვას).

ამ ეტაპზე წარმოვადგენთ რამოდენიმე ფაქტს, თუ რატომ ავირჩიეთ დაპროგრამების ენა C, იმის სანაცვლოდ, რომ მიგვემართა მაღალი დონის უფრო ძლიერი პროგრამული უზრუნველყოფისთვის [1]:

- ვისწავლით რა მოდელირებას C უნივერსალურ ენაზე, ყოველი დეტალი, რომელსაც ესაჭიროება განსაკუთრებული ყურადღება, შეიძლება უკეთ გავიგოთ მოდელის მოქმედებაში. მაშასადამე რჩება ნაკლები შანსი დავუშვათ კონცეპტუალური შეცდომები, თუ გადავალთ მაღალი დონის პროგრამულ უზრუნველყოფაზე;
- მიუხედავად იმისა, რომ არსებობს სიმულაციური მოდელირების სპეციალური მძლავრი პროგრამული უზრუნველყოფა (მიმოხილვა იხ. §5.5), რთული სისტემების მოდელირებისას ხშირად მაინც გვიწევს დაპროგრამების ნაწილი შესრულდეს უნივერსალურ ენაზე;
- მოდელირების უნივერსალური ენები ფართოდ ხელმისაწვდომია, მასზე დღეს-დღეობით წერენ მნიშვნელოვან სიმულაციურ მოდელებს;
- აღნიშნული კურსი დაწერილია დაბალი კურსის სტუდენტებისთვის (უფრო III კურსი). მათ კი ენა C -ს ცოდნა მოეთხოვებათ სამომხმარებლო დონეზე.

ლექციათა კურსში ჩვენი მიზანი არაა დაწვრილებით განვიხილოთ სიმულაციური მოდელირების რომელიმე კონკრეტული პროგრამული უზრუნველყოფა (თუმცა გარკვეული პაკეტების მიმოხილვას გავეცნობით მოგვიანებით). ცოდნა, რომელსაც თქვენ შეიძენთ განზოგადოებული მიდგომებისა და სისტემების მოდელირების მაგალითების გაცნობით, დაგეხმარებათ იოლად აითვისოთ მოდელირების სპეციალიზირებული პროგრამული უზრუნველყოფები. მმსმეკ, რომლის მოდელირებასაც უკვე პროგრამულად გავაკეთებთ, განსხვავდება წინა პარაგრაფში განხილული მაგალითისგან ორი შემდეგი ფაქტორით:

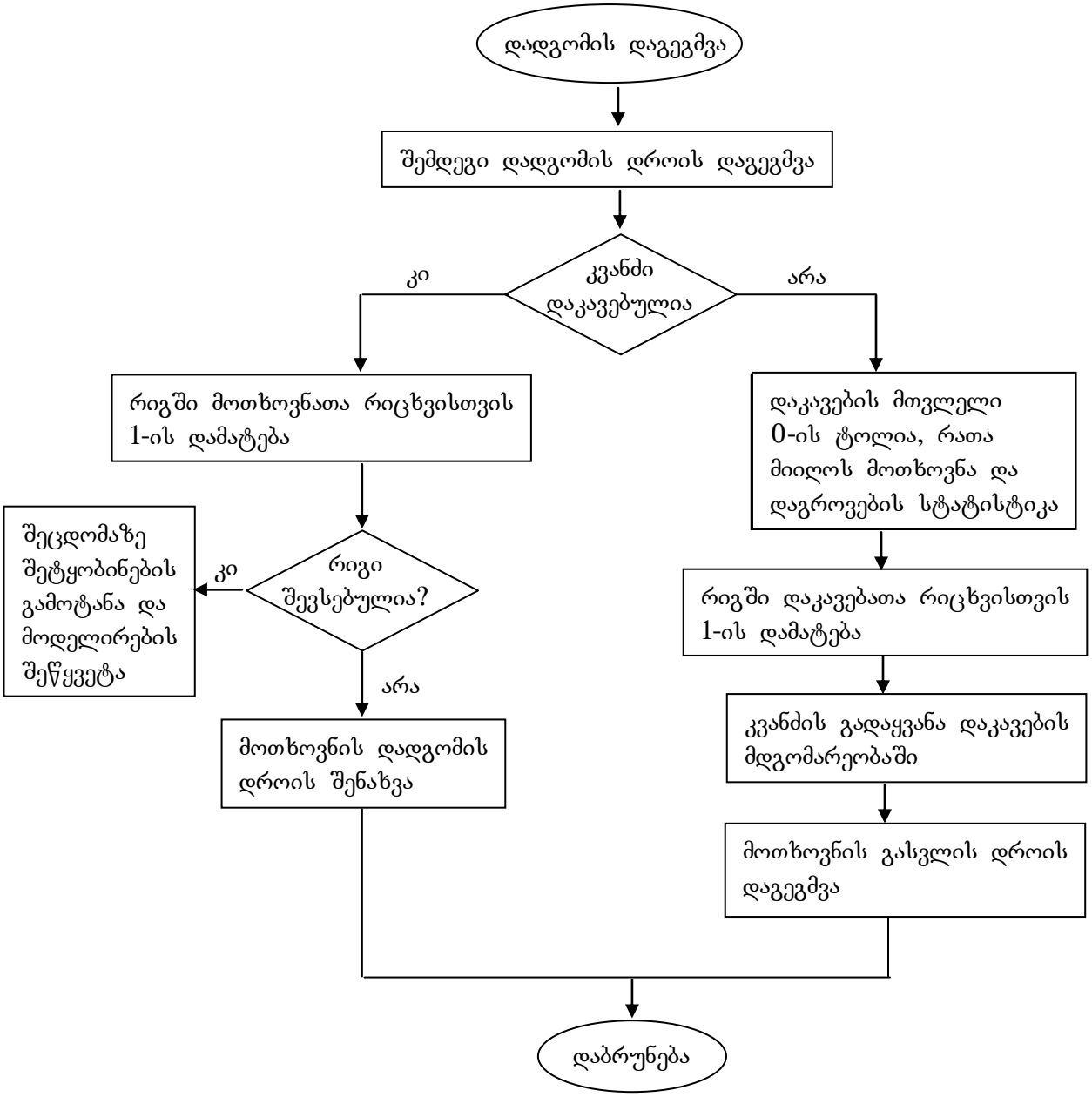
- *მოდელირება დასრულდება* მაშინ, როდესაც რიგში დაკავებულ მოთხოვნათა რიცხვი მიაღწევს $n = 1000$ და არა $n = 6$, რაც დააგროვებს სრულ სტატისტიკას სისტემის მუშაობის შეფასებისთვის, რამეთუ იცვლებიან $\hat{d}(n)$, $\hat{q}(n)$ და $\hat{u}(n)$ და ისინი მდგრადი შეფასებები ხდება n-ის მიმართ.
- *მოთხოვნების დადგომებს შორის დრო* და *მომსახურეობის დრო* აქ უკვე გენერირდება ექსპონენციალური განაწილებიდან, როგორც დამოუკიდებელი შემთხვევითი სიდიდეები მოთხოვნების დადგომებს შორის საშუალო $\frac{1}{\lambda_A} = 1$ წთ დროის ინტერვალითა და მომსახურეობის საშუალო $\frac{1}{\lambda_D} = 0{,}5$ წთ დროის ინტერვალით. λ ინტენსივობის მქონე (საშუალო დროის შებრუნებული სიდიდე) ექსპონენციალური განაწილება, როგორც ვიცით უწყვეტი განაწილებაა:

$$f(x) = \begin{cases} \lambda\, e^{-\lambda x}, & x > 0, \\ 0, & x \le 0, \end{cases} \qquad F(x) = \int_{-\infty}^{x} f(t)\, dt = \begin{cases} 1 - e^{-\lambda x}, & x > 0, \\ 0, & x \le 0. \end{cases}$$

მოთხოვნების დადგომის შორის დრო და მომსახურეობის დრო უმეტესწილად გენერირდება ექსპონენციალური განაწილების გენერატორიდან (იხ. მაგ. 3.1), მისი კომპიუტერზე გენერირების ალგორითმი მარტივია. შევნიშნოთ, რომ მოთხოვნებს შორის დრო როგორც ყოველთვის ექსპონენციალური განაწილებაა, რასაც ვერ ვიტყვით მომსახურეობის დროზე, რომელიც სხვადასხვა მაგალითებში განსხვავებულია. საჭირო იქნება შესაბამისი გენერატორის შექმნა.

მასობრივი მომსახურეობის სისტემა, მოთხოვნათა დადგომებს შორის დროისა და მომსახურეობის დროის ექსპონენციალური განაწილებით, ტრადიციულად $M/M/1$ სიმბოლოთი აღინიშნება (მასობრივი მომსახურეობის სისტემათა კლასიფიკაციას გავეცნობით მოგვიანებით).

$M/M/1$ მოდელის ასაგებად ჩვენ გვჭირდება ექსპონენციალური განაწილების გენერატორი. თუ გვაქვს γ შემთხვევითი რიცხვის გენერატორი $\gamma \sim U(0,1)$, მაშინ $F(X)=\gamma$ განტოლების ამოხსნით მივიღებთ ექსპონენციალურ გენერატორს $X=-\frac{1}{\lambda}\ln\gamma$.



ნახ. 4.5

$M/M/1$-ის პროგრამა მოსახერხებლია შევკრათ რამოდენიმე ქვეპროგრამა ბლოკებად, ფუნქციებად. ეს გვეხმარება, რათა მისი ლოგიკა და კომპონენტების ურთიერთქმედება გახდეს უფრო გასაგები. ძირითადი პროგრამის გარდა მოდელირების პროგრამა ითვალისწინებს *ინიციალიზაციის, სინქრონიზაციის, ანგარიშების გენერატორის, ხდომილებათა დამუშავებისა და ექსპონენციალური გენერატორის ქვეპროგრამათა ფუნქციების* ჩართვას (იხ. ნახ. 1.3).

უწყვეტი დროის $\hat{Q}(t)$ და $\hat{B}(t)$ სტატისტიკების განახლებისთვის მოდელირების პროცესში გაცილებით იოლია ცალკე ფუნქციების დაწერა. ყველაზე მთავარი მოქმედება ხდება ხდომილებათა დამუშავების ფუნქციებში, რომელთაც შემდეგი ნუმერაცია აქვს:

ხდომილება	ხდომილების ტიპი
მოთხოვნის შემოსვლა სისტემაში	1
მოსახურეობის დასასრულს მოთხოვნის გასვლა სისტემიდან:	2

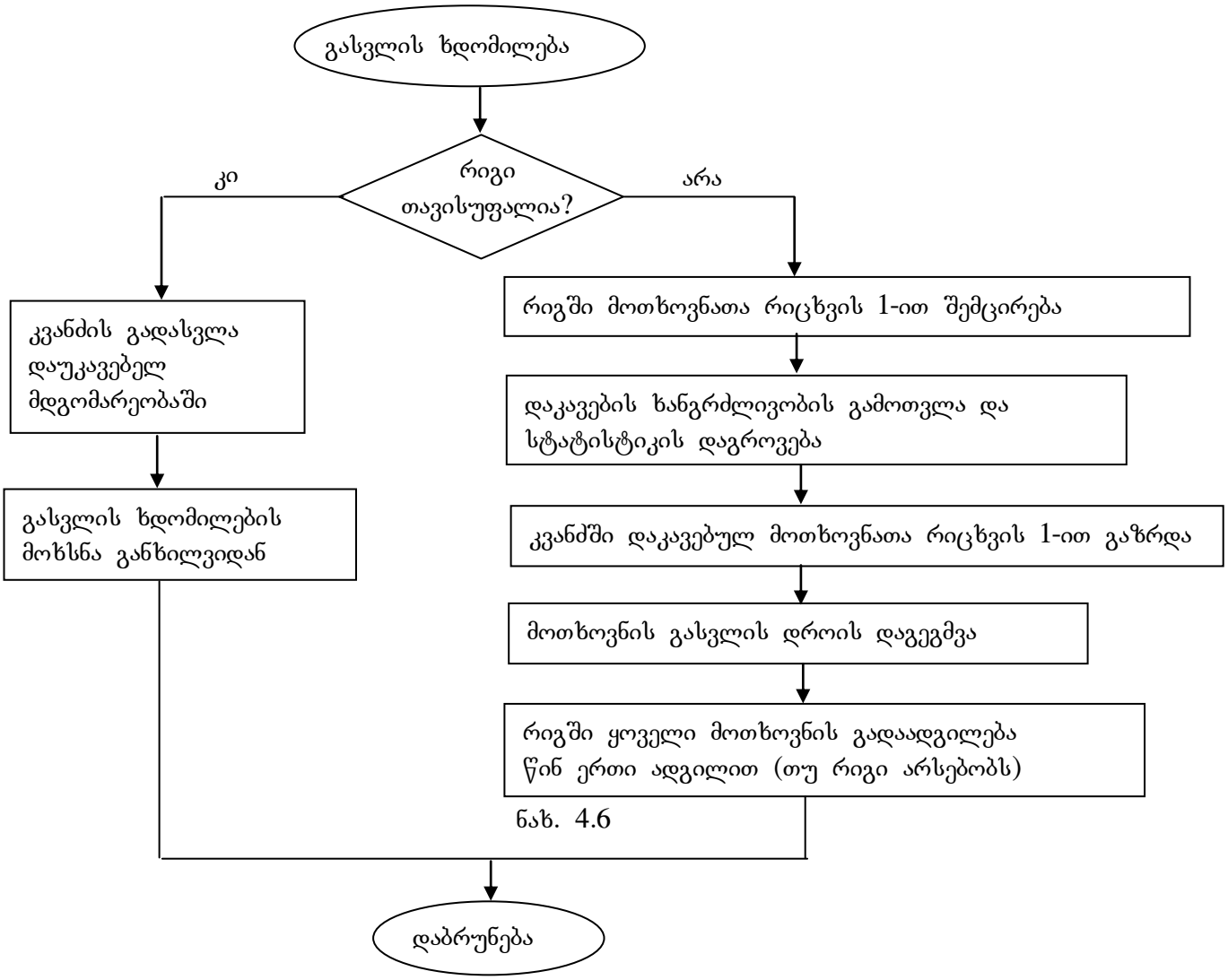
რადგან ხდომილების დამუშავების ლოგიკა არაა დამოკიდებული იმ ენაზე, რომელიც გამოიყენება დაპროგრამებისას, განვიხილოთ იგი ამ პარაგრაფში. ნახ. 4.5-ზე წარმოდგენილია მოთხოვნის დადგომის ბლოკსქემა.

პირველ რიგში გენერირდება შემდეგი დადგომის დრო და იგი ჩაიწერება ხდომილებათა სიაში. შემდეგ მოწმდება კვანძის დაკავებულობა. როდესაც კვანძში დაკავებულია, რიგში მოთხოვნათა რიცხვი იზრდება 1-ით, და ჩვენ ვამოწმებთ შევსებულია თუ არა მეხსიერება გამოყოფილი რიგის შესანახად. თუ რიგი შევსებულია, მაშინ გენერირდება შეტყობინება შეცდომაზე და მოდელირება წყდება. თუ რიგში კიდევაა ადგილი, მაშინ მოთხოვნის დადგომის დრო მოთავსდება რიგის ბოლოში (ასეთი სრული შემოწმება რიგის გადავსებაზე შეიძლება მოიხსნას, თუ გამოვიყენებთ დაპროგრამების ენაში მეხსიერების დინამიკურ განაწილებას). თუ მოთხოვნის დადგომისას კვანძი თავისუფალია, მაშინ მოთხოვნის დაგვიანება რიგში 0-ის ტოლი იქნება, მაგრამ მაინც ითვლება დაგვიანებად და მოთხოვნათა დაგვიანების რიცხვი 1-ით იზრდება! კვანძი გადადის დაკავებულ მდგომარეობაში, ხოლო ხდომილებათა სიაში იგეგმება მომსახურეობის დასრულების შემდეგ გასვლის დრო.

მოთხოვნის გასვლის პროგრამის ლოგიკა წარმოდგენილია ნახ. 4.6-ზე. როგორც გახსოვთ ეს პროგრამა გამოიძახება მოთხოვნის მომსახურეობის დასრულებისთანავე.

თუ მოთხოვნის გასვლის შემდეგ რიგი რჩება ცარიელი, მაშინ კვანძი გადადის დაუკავებელ მდგომარეობაში, ხოლო მოთხოვნის გასვლის აღრიცხვა გადაიდება, რადგან შემდეგი ხდომილება უნდა იყოს მოთხოვნის დადგომა. თუ მოთხოვნის გასვლის შემდეგ რიგში რჩება ერთი ან რამოდენიმე მოთხოვნა, პირველი მოთხოვნა რიგიდან ტოვებს რიგს და გადადის მომსახურეობაზე. ამ დროს რიგის სიგრძე 1-ით მცირდება, ხოლო ამ მოთხოვნის დაკავების ხანგრძლივობა გამოითვლება და დარეგისტრირდება შესაბამისი სტატისტიკური მთვლელით. რიგში დაკავებულთა რიცხვი 1-ით იზრდება და იგეგმება მოთხოვნის გასვლის დრო, რომელიც გადავიდა მომსახურეობაზე. ამის შემდეგ რიგში დარჩენილი მოთხოვნები (თუ ასეთი არსებობს) გადაინაცვლებენ ერთი ადგილით წინ. ამ თავში ნაჩვენები რიგის სიის რეალიზაციის ხერხი ძალიან მარტივია, თუმცა არც თუ ეფექტურია. არსებობს მოდელირებისას მიღებული სიების დამუშავების უფრო ეფექტური და მოსახერხებელი ალგორითმები [1].

შემდეგ პარაგრაფში ნაჩვენებია, თუ როგორ უნდა გამოვიყენოთ აქ წარმოდგენილი საკითხები C ენაზე მოდელირების პროგრამის დაწერისას. ჩაწერილი პროგრამული კოდები[1] არაა არც ისე მარტივი, არც ისე ეფექტური. ისინი შექმნილია და მოყვანილია იმისთვის, რომ ვაჩვენოთ სამომავლოდ, თუ როგორ შეიძლება *შევქმნათ მაღალ-ორგანიზებული კომპიუტერული სიმულაციური მოდელირება* უფრო რთული სისტემებისთვის.



ნახ. 4.6

§4.4. მმსმეკ-ის პროგრამა C-ზე

ამ პარაგრაფში წარმოდგენილია C-ზე დაწერილი პროგრამა $M/M/1$ მასობრივი მომსახურეობის სისტემის მოდელირებისთვის (ყველა პროგრამული კოდი შეიძლება თქვენს კომპიუტერზე ჩაიტვირთოს www.mhhe.com/lawkelton ელ-გვერდიდან (მონაცემები წარმოდგენილია [1]-დან)). შევნიშნოთ, რომ პროგრამის რიცხვითი შედეგები ოდნავ შეიძლება განსხვავდებოდეს სხვადასხვა კომპიუტერსა და კომპილიატორზე. პროგრამებთან მიერთებული უნდა იყოს C ენის მათემატიკური ბიბლიოთეკა, რაც მოითხოვს კომპილიატორზე დამოკიდებული ოფციების დაყენებას (UNIX სისტემაში ეს ოფციაა – em კომპილაციის ოპერატორში).

ლისტინგ 4.1-ზე მოცემულია გარე განმსაზღვრელები. დასათაურების ფაილი lcgrand.h (იხ. ლისტინგი 2.2) ჩართულია შემთხვევითი რიცხვების გენერატორის ფუნქციათა განაცხადში. Q_LIMIT სიმბოლურ კონსტანტას მიენიჭება მნიშვნელობა – 100, შესაძლო მაქსიმალური სიგრძე რიგში. BUSY და IDLE სიმბოლური მუდმივები განისაზღვრება server_status (კვანძის მდგომარეობა) ცვლადის გამოყენებისთვის. იმისთვის, რომ კოდი გახდეს უფრო მარტივად წასაკითხი, განსაზღვრულია ფაილის მაჩვენებლები *infile და *outfile, რომელიც საშუალებას მოგვცემს გავხსნათ შესაბამისად შესავალი და გამოსავალი ფაილები კოდიდან და არა სისტემის ფუნქციონირების დონიდან. მივაქციოთ ყურადღება იმას, რომ ხდომილებათა სია, რაზეც ადრე იყო საუბარი, შესრულდება როგორც მასივი time_next_event ნულოვანი

უჯრედის იგნონირებით, რათა ინდექსი შეესაბამებოდეს ხდომილების ტიპს. მდს-ს მნიშვნელობა ინახება sim_time ცვლადში.

ლისტინგი 4.1 (კოდი C-ზე გარე განსაზღვრებებისთვის).

```
/* M/M/1 მოდელის გარე განსაზღვრებანი */
#include <stdio.h>
#include <math.h>

/* დასათაურების ფაილი შემთხვევითი რიცხვების გენერატორისთვის */
#include "lcgrand.h"

#define Q_LIMIT 100  /* რიგის სიგრძის შეზღუდვა */
#define BUSY 1    /* კვანძის მდგომარეობის მნემონიკური კონსტანტები */
#define IDLE 0    /* მისი დაკავების ან თავისუფლების განზღვრისათვის */

int next_event_type, num_cust_delayed, num_delays_required, num_events,
      num_in_q, server_status;
float area_nam_in_q, area_server_status, mean_interarrival, mean_service, sim_time,
      time_arrival [Q_LIMIT+1], time_last_event, time_next_event [3], total_of_delays;
FILE  *infile, *outfile;
void  initialize (void);
void   timing (void);
void   arrive (void);
void   depart (void);
void   report (void);
void   update_time_arg_stats (void);
float  expon (float mean);
```

ძირითადი ფუნქციის კოდი ნაჩვენებია ლისტინგ 4.2-ზე. მასში იხსნება შემავალი და გამოსავალი ფაილები. სისტემის მოდელირებისას ხდომილებათა ტიპების რიცხვი *ყენდება* 2-ზე. შემდეგ *შემავალი პარამეტრები იკითხება mm1.in ფაილიდან*, რომელიც შეიცავს ერთ სტრიქონს, პრობელით გამოყოფილ სამ რიცხვს: 1,0; 0,5 და 1000. რომ შემოწმდეს სწორადაა თუ არა ჩაწერილი ეს სიდიდეები ფაილში, გამოიძახება ინიციალიზაციის ფუნქცია. ციკლი while მოდელირებას ასრულებს მანამ, სანამ არ დარეგისტრირდება მე-1000 დაკავება რიგში (გაჩერების წესი). ციკლ while-ში თავიდან გამოიძახება სინქრონიზაციის ფუნქცია, რომელიც განსაზღვრავს შემდეგი ხდომილების ტიპს და მდს გადაყავს ხდომილების დადგომის დროზე. მოცემული ხდომილების დამუშავების წინ გამოიძახება ფუნქცია, რომელიც განაახლებს $Q(t)$ და $B(t)$ წირების ქვემოთ ფართობებს. ანუ ჩვენ ავტომატურად განვაახლებთ ამ ფართობებს ყოველი ხდომილების დამუშავების წინ. შემდეგ switch ოპერატორი, რომელიც იყენებს next_event_type (ტოლია 1-ის, როცა მოთხოვნა შემოდის; 2-ის, როცა მოთხოვნა გადის) ცვლადს და მართვას გადასცემს შესაბამისი ხდომილების დამუშავების ფუნქციას. while ციკლის შესრულების შემდეგ გამოიძახება report ფუნქცია (ანგარიშების ფუნქცია), შემავალი და გამომავალი ფაილები იხურება და მოდელირება სრულდება.

ლისტინგი 4.2 (კოდი C-ზე ძირითადი ფუნქციისთვის).

```
main ( ) /* ძირითადი ფუნქცია */
{
      /* ვხსნით შესავალ და გამოსავალ ფაილებს */
      infile = fopen ("mm1.in", "r");
      outfile = fopen ("mm1.out", "w");
      /* სინქრონიზაციის ფუნქციისთვის მივუთითებთ ხდომილებათა რიცხვს */
      num_event = 2;
```

```
        /* ვკითხულობთ შემავალ პარამეტრებს */
        fscan f (infile, "%f %f %d", &mean_interarrival,
                & mean_service, &num_delays_required);
        /* მივუთითებთ ანგარიშის სათაურს და შემავალ პარამეტრებს */
        fprintf (outfile, "მასობრივი მომსახურეობის სისტემა ერთი კვანძით \n\n");
        fprintf (outfile, "დადგომებს შორის საშუალო დრო %11.3 f წთ \n\n", mean_interattival);
        fprintf (outfile, "მომსახურეობის საშუალო დრო %16.3 f წთ \n\n", mean_service);
        fprintf (outfile, "მოთხოვნათა რიცხვი %14 d\n\n", num_delays_required);
        /* მოდელირების ინიციალიზაცია */
        initialize ( );
        /* ვასრულებთ მოდელირებას, სანამ არ დარეგისტრირდება */
        /* რიგში მოთხოვნათა დაკავების საჭირო რაოდენობა */
        while (num_custs_delayed <num_delays_required) {
                /* განვსაზღვრავთ შემდეგ ხდომილებას */
                timing ( );
                /* განვაახლებთ საშუალო სტატისტიკურ დამგროვებლებს */
                update_time_avg_stats ( );
                /* ვიძახებთ შესაბამისი ხდომილების დამუშავების ფუნქციას */
                switch (next_event_type) {
                        case 1:
                                arrive ( );
                                break;
                        case 2:
                                depart ( );
                                break
                }
        }
        /* ვიძახებთ ანგარიშების გენერატორს და ვასრულებთ მოდელირებას */
        report ( );
        fclose (infile);
        fclose (outfile);
        return 0;
}
```

ინიციალიზაციის ფუნქციის კოდი წარმოდგენილია ლისტინგ 4.3-ზე. მისი ყოველი ოპერატორი შეესაბამება კომპიუტერულ წარმოდგენას ნახ. 4.4ა-დან. შევნიშნოთ, რომ პირველი დადგომის დრო time_next_event[1] განისაზღვრება ექსპონენციალურად განაწილებული შემთხვევითი სიდიდისთვის დადგომებს შორის საშუალო მნიშვნელობის mean_interarrival მიმატებით. ანუ ექსპონენციალურ განაწილებას ითვლის ფუნქცია expon (mean_interarrival) და მას მიუმატებს მდს-ის, sim_time-ის მნიშვნელობას. რადგან დროის მომენტში, როდესაც sim_time = 0, სისტემაში არაა მოთხოვნა, შემდეგი გასვლის დრო time_next_event[2] ყენდება 1.0e+30-ზე (C ენაზე 10^{30} ასე აღინიშნება) და ესაა გარანტი იმისა, რომ შემდეგი ხდომილება იქნება მოთხოვნის დადგომა.

ლისტინგი 4.3 (კოდი C-ზე, ინიციალიზაციის ფუნქცია).

```
Void initialize (void)  /* ინიციალიზაციის ფუნქცია */
{
        /* მდს-ს ვაყენებთ საწყის მდგომარეობაში */
        sime_time = 0.0;
        server_status    = IDLE;
        num_in_q         = 0;
        time_last_event = 0.0;
        /* სტატისტიკურ მთვლელებს ვაყენებთ საწყის მდგომარეობაში */
        num_cust_delayed  = 0;
```

```
        total_of_delays      = 0;
        area_num_in_q        = 0.0;

        /* ხდომილებათა სიის ინიციალიზაცია. გასვლის აღრიცხვის მოხსნა */
        /* (მომსახურეობის დასრულება), რადგან სისტემაში არაა მოთხოვნა */
        time_next_event[1] = sim_time + expon (mean_interarrival);
        time_next_event[2] = 1.0e + 30;
}
```

სინქრონიზაციის ფუნქცია (ლისტინგი 4.4) გამოიყენება იმისთვის, რომ შევადაროთ ხდომილებათა წარმოქმნის დროები time_next_event[1], time_next_event[2],..., time_next_ event [num_events] (ცვლადი num_events განსაზღვრული იყო ძირითად ფუნქციაში) და დავადგინოთ შემდეგი ტიპის ხდომილება – num_events, რომელსაც გააჩნია წარმოქმნის (დადგომის) უმცირესი დრო. *დროითი კავშირის* წარმოშობის შემთხვევაში აირჩევა იმ ტიპის ხდომილება, რომელსაც გააჩნია ნაკლები ნომერი. ამის შემდეგ მოდელური დროის საათი გადაიყვანება min_time_next_event – არჩეული ტიპის ხდომილების წარმოშობის დროზე. პროგრამას რამდენადმე ართულებს შემოწმების აუცილებლობა – ხომ არაა ხდომილებათა სია ცარიელი? ამასთან ერთად ჩვენ გამოვდივართ იქიდან, რომ ყველა ხდომილება უნდა დადგეს 10^{30} მომენტამდე. თუ ასეთი სიტუაცია წარმოიშობა, რომელზეც მიუთითებს next_event_type = 0, გენერირდება შეტყობინება შეცდომაზე მიმდინარე დროსთან ერთად და მოდელირება წყდება.

ლისტინგი 4.4 (კოდი C-ზე, სინქრონიზაციის ფუნქცია).

```
Void timing (void)     /* სინქრონიზაციის ფუნქცია */
{
        int i;
        float min_time_next_event =1.0 e + 29;
        next_event_type = 0;
        /* განვსაზღვრავთ შემდეგი ხდომილების ტიპს */
        for (i = 1;  <i<=num_events; ++ i)
                if (time_nest_event[i] < min_time_nest_event) {
                    min_time_next_event = time_next_event [i];
                    next_event_type =i;
                }
        /* ვამოწმებთ, არის თუ არა ხდომილებათა სია ცარიელი */
        if (next_event_type = = 0) {
                /* ხდომილებათა სია ცარიელია, ვამთავრებთ მოდელირებას */
                fprint f (outfile, "\n ხდომილებათა სია ცარიელია მომენტში %f", sim_time);
                exit(1);
        }
        /* ხდომილებათა სია არაა ცარიელი. მდს გადაგვყავს */
        sim_time = min_time_next_event;
}
```

ხდომილებათა დამუშავების ფუნქციის კოდი arrive (ლისტინგი 4.5) აგებულია ბლოკ-სქემის (ნახ. 4.5) მიხედვით. მივაქციოთ ყურადღება იმას, რომ sim_time – ეს დროა, როდესაც ხდომილება დადგება (წარმოიქმნება) მოცემულ მომენტში და რომ სრულდება შემოწმება რიგის გადავსებაზე. ამასთან ერთად ვკითხულობთ, იქნება თუ არა num_in_q-ს მნიშვნელობა მეტი Q_LIMIT ცვლადის მნიშვნელობაზე, რომელიც განსაზღვრავს time_arrival მასივის სიგრძეს.

ლისტინგი 4.5 (კოდი C-ზე, ფუნქცია arrive)

```
void arrive (void)  /* ხდომილებითი ფუნქცია მოთხოვნის დადგომაზე */
{
      float delay;
      /* ვგეგმავთ შემდეგ დადგომას */
      time_next_event[1] = sim_time + expon (mean_interarrival);
      /* ვამოწმებთ, დაკავებულია თუ არა მომსახურეობის კვანძი */
      + + num_in_q;
      /* ვამოწმებთ გადავსების პირობას */
      if (num_in_q > Q_LIMIT) {
            /* რიგი გადავსებულია, მოდელირება წყდება */
            fprintf (outfile, "\n მასივის გადავსება – time_arrival");
            fprintf (outfile, "% f – დროის მომენტში," sim_time);
            exit (2);
            }
      /* რიგში კიდევ რჩება ადგილი, ვინახავთ */
      /* მოთხოვნის დადგომის დროის time_arrival მასივის ბოლოში */
      time_arrival [num_in_q] = sim_time;
   }
   else {
      /* კვანძი თავისუფალია, რიგში შემოსული მოთხოვნის დაკავება ნულის ტოლია */
      /* (შემდეგი ორი ოპერატორი არ მოქმედებს მოდელირების შედეგებზე) */
      delay = 0.0
      total_of_delays + = delay;
      /* მოთხოვნათა დაკავების რიცხვს ვზრდით  და კვანძი გადაგვყავს */
      /* დაკავების მდგომარეობაში */
      + + num_cust_delayed;
      server_status = BUSY;
      /* ვგეგმავთ მოთხოვნის გასვლას (ვამთავრებთ მომსახურეობას) */
      time_next_event[2] = sim_time + expon (mean_setvice);
      }
}
```

ხდომილებათა ფუნქცია depart (ლისტინგი 4.6) გამოიძახება მაშინ, როდესაც ხდება მოთხოვნის მომსახურეობის დასრულება (და მისი შემდგომი გასვლა სისტემიდან). ამ ფუნქციის ლოგიკა განხილული იყო წინა პარაგრაფში და აისახა ბლოკ-სქემაზე (ნახ. 4.6). თუ ოპერატორს time_next_event[2] = 1,0 e + 30 ოპერატორი else-ს წინ მოვაცილებთ, მაშინ პროგრამა შევა უსასრულო ციკლში. ეს მოხდება იმიტომ, რომ რიგის დარჩენილი ნაწილის წინ წანაცვლება (თუ ასეთი არსებობს) for ციკლის საშუალებით გარანტიას იძლევა, რომ მოთხოვნის დადგომის დრო, რომელიც შემდეგ შემოვა მომსახურეობაზე, ყოველთვის შეინახება time_arrive[1] მასივში. თუ რიგი თავისუფალი იქნება, მაშინ num_in_q ცვლადი უტოლდება 0-ს, ხოლო ციკლი საერთოდ არ შესრულდება, რადგან ციკლის i ინდექსის საწყისი მნიშვნელობაა i = 1, რომელიც სცილდება მის ბოლო მნიშვნელობას. უკანასკნელი შენიშვნა depart ფუნქციასთან მიმართებაში ეხება time_arrival[1]-ის გამოკლებას sim_time საათის ჩვენებიდან, რათა მივიღოთ რიგში დაკავების დროის მნიშვნელობა. თუ მოდელირება მიმდინარეობს დიდი დროის განმავლობაში, sim_time და time_arrival[1] ღებულობენ დიდ მნიშვნელობებს მათ სხვაობასთან შედარებით. რადგან ისინი დაიმახსოვრებიან, როგორც რიცხვები მცოცავი მძიმით (float), მათი სიზუსტე შეიძლება არ აღმოჩნდეს საკმარისი და შეიძლება წარმოიშვას დანაკარგები გამოთვლების ჩატარებისას. აქედან გამომდინარე, თუ ჩვენ გვინდა, რომ მოდელირება გრძელდებოდეს დიდი დროით, უნდა მივანიჭოთ double ტიპი sim_time ცვლადისა და time_arrival მასივისათვის.

ლისტინგი 4.6 (კოდი C-ზე, ფუნქცია depart)

```
Void depart (void)   /* ხდომილებითი ფუნქცია მოთხოვნის გასვლისთვის */
{
        int i;
        float delay;
        /* ვამოწმებთ, არის თუ არა რიგში მოთხოვნა */
        if (num_in_q = = 0)  {
                /* რიგში არაა მოთხოვნა; კვანძი გადაგვყავს თავისუფალ მდგომარეობაში */
                /* და მოვხსნით გასვლის აღრიცხვას (მომსახურეობის დასრულება). */
                server_status          = IDLE;
                time_next_event[2] = 1.0e + 30;
        }
        else  {
                /* რიგში არის მოთხოვნა. ვამცირებთ რიგში მოთხოვნათა რიცხვს */
                - - num_in_q;
                /* ვითვლით დაკავებას მოთხოვნათა რიგში, რომელთა მომსახურეობა დაიწყო,
                და განვაახლებთ დაკავების საერთო დამგროვებლის მნიშვნელობას */
                delay                  = sim_time-time_arrival[1];
                total_of_delays + = delay;
                /* ვზრდით მოთხოვნების დაკავებათა რიცხვს და ვგეგმავთ გასვლის დროს */
                + + num_cust_delayed;
                time_next_event[2] = sim_time + expon (mean service);
                /* გადაგვყავს რიგში ყოველი მოთხოვნა ერთი პოზიციით წინ */
                for  (i = 1; < <= num_in_q; ++i)
                        time_arrival[i] = time_arrival[i+1]
                }
}
```

report ფუნქციის კოდს ვიძახებთ ციკლი while-ის შესრულების შემდეგ ძირითად პროგრამაში (ლისტინგი 4.7). *საშუალო დაკავებას* ვითვლით საერთო დროის შეფარდებით მოთხოვნათა რიცხვზე, რომელთა დაკავება რიგში დაფიქსირდა. *მოთხოვნათა საშუალო რიცხვს*, რომელიც იმყოფება რიგში დროის ერთეულ მომენტში, ვღებულობთ $Q(t)$ წირით ქვემოთ შემოსაზღვრული ფართობის შეფარდებით მოდელირების დასრულების მომენტის მნიშვნელობაზე. *კვანძის დაკავებულობის კოეფიციენტი* გამოითვლება $B(t)$ წირით ქვემოთ შემოსაზღვრული ფართობის შეფარდებით მოდელირების დროის საბოლოო მნიშვნელობაზე. სამივე მახასიათებელი ჩაიწერება ანგარიშში. გარდა ამისა, ჩაიწერება *მოდელირების დროის საბოლოო მნიშვნელობაც*, რომ გავიგოთ თუ რა დრო შეიქმნა საჭირო რიგში 1000 მოთხოვნის დაკავების ფიქსაციისთვის!

ლისტინგი 4.7 (კოდი C-ზე, ფუნქცია report)

```
Void report (void)   /* ანგარიშების გენერატორის ფუნქცია */
{
        /* ვითვლით და ვიმახსოვრებთ შეფასების კრიტერიუმებს */
        fprintf (outfile, "\n\n რიგში საშუალო დაკავება % 11.3 f წთ \n\n",
                total_of_delays / num_custs_delayed);
        fprintf (outfile, "რიგში მოთხოვნათა საშუალო რიცხვი % 10.3 f \n\n",
                area_num_in_q / sim_time);
        fprintf (outfile, "მომსახურეობის კვანძის დაკავების კოეფიციენტი % 15.3 f \n\n",
                area_server_status / sim_time);
        fprint (outfile, "მოდელირების დასრულების დრო % 12.3.f  წთ", sim_time);
}
```

ფუნქცია update_time_avg_stats (ლისტინგი 4.8) გამოიძახება ნებისმიერი ტიპის ხდომილების დამუშავების წინ. ის განაახლებს ფართობის მნიშვნელობებს ორი $Q(t)$ და $B(t)$ ფუნქციათა წირების ქვემოთ, რათა მივიღოთ უწყვეტი ტიპის სტატისტიკები სისტემის მუშაობის შეფასებისთვის. ეს პროგრამა *არ არის* ხდომილების დამუშავების პროგრამა და გამოყოფილია მხოლოდ იმისთვის, რომ კოდის დაწერა უფრო მოსახერხებელი შეიქმნას. თავიდან გამოითვლება დრო, რომელმაც გაიარა ბოლო ხდომილების დადგომის შემდეგ. უკანასკნელი ხდომილების დრო შეიცვლება მიმდინარე დროით ამ ფუნქციაში შემდეგი შესვლის მოსამზადებლად. შემდეგ $Q(t)$ ფუნქციის ქვევით წირით შემოსაზღვრული ფართობი, რომელიც ითვლის რიგში მოთხოვნათა რიცხვს, იზრდება იმ ფართობით $Q(t)$ -ს წირის ქვევით, რომელიც გამოითვლება time_since_last_event – ბოლო ხდომილების დადგომის დროის შემდეგი შუალედისა და num_in_q სიდიდის ნამრავლით. არ დაგვავიწყდეს, რომ ეს ფუნქცია გამოიძახება *ხდომილების დამუშავებამდე* და ისეთი ცვლადები, როგორიცაა num_in_q, ინარჩუნებს თავის ძველ მნიშვნელობას. $B(t)$ ფუნქციის წირის ქვევით ფართობი იზრდება მართკუთხედის ფართობით, რომლის სიგრძეა time_since_last_event, ხოლო სიმაღლეა server_status, რომელიც ან 0-ის ან 1-ის ტოლია. ეს ფუნქცია, depart ფუნქციის მსგავსად ახდენს ორი რიცხვის გამოკლებას მცოცავი მძიმით (sim_time – time_last_event), რომელმაც ხანგრძლივი მოდელირებისას შეიძლება მიაღწიოს დიდ სიდიდეებს. ამ სიტუაციაში შეიძლება საჭირო შეიქმნას sim_time-ის და time_last_event ცვლადები გამოვაცხადოთ, როგორც double.

ლისტინგი 4.8 (კოდი C-ზე, ფუნქცია update_time_avg_status)

```
Void update_time_avg_status (void) /* საშუალო დროის სტატისტიკების – ფართობების
                                      გამოთვლის დამგროვებელის განახლება */
{
    float time_since_last_event;
    /* ვითვლით დროს, რომელმაც გაიარა ბოლო ხდომილების დადგომის შემდეგ
       და ვანაახლებთ ბოლო ხდომილების მარკერს */
    time_since_last_event = sim_time – time_last_event;
    time_last_event       = sim_time;
    /* რიგში მოთხოვნათა რიცხვის განსაზღვრისათვის ვანაახლებთ ფართობს
       ფუნქციის წირის ქვევით */
    area_num_in_1      + = num_in_q * time_since_last_event;
    /* კვანძის დაკავებულობის კოეფიციენტის განსაზღვრისათვის ვანახლებთ
       ფართობს ფუნქციის წირის ქვევით */
    area_server_status  + = server_status * time_since_last_event;
}
```

expon ფუნქცია (ლისტინგი 4.9), რომელიც გენერირებას უკეთებს ექსპონენციალურ განაწილებას $\beta = \text{mean}\left(\beta = \frac{1}{\lambda}\right)$ საშუალო მნიშვნელობით, შეესაბამება ალგორითმს, რომელიც განვიხილეთ §3.1.2-ში. შემთხვევით რიცხვთა გენერატორი lcgrand, რომელიც იყენებს int არგუმენტს (int = 1) დეტალურადაა განხილული ლისტინგ 2.1-ზე. log ფუნქცია C-ზე აბრუნებს არგუმენტის ნატურალურ ლოგარითმს.

ლისტინგი 4.9 (კოდი C-ზე, ფუნქცია expon)

```
float expon (float mean)     /* ექსპონენციალურად განაწილებული შემთხვევითი
                                სიდიდის გენერირების ფუნქცია */
{
    /* ვაბრუნებთ ექსპონენციალურ შემთხვევით სიდიდეს საშუალოთი mean */
    return-mean * log (lcgrand(1));
}
```

აღწერილი კოდი ურთიერთქმედებს lcgrand კოდთან, გენერატორთან, რომლის ლისტინგი 2.1 მოცემული თავ 2-ში. ეს რომ უზრუნველვყოთ, შეიძლება გავაკეთოთ *თითოეული პროგრამის ცალკე კომპილაცია.*

§4.5. სიმულაციური მოდელირების გამოსავალი მონაცემები და მათი სტატისტიკური ანალიზი

გამოსავალი მონაცემები mm1.out ფაილიდან ნაჩვენებია ლისტინგ 4.10-ზე. როგორც გამოთვლები გვიჩვენებს (აღნიშნული გენერატორის გამოყენებით) მოდელირების ამ გატარებაში *რიგში მოთხოვნის დაკავების საშუალო დრო* შეადგინა 0.430 წთ, საშუალოდ რიგში იმყოფებოდა 0,418 მოთხოვნა (წთ-ში), *კვანძის დაკავებულობამ შეადგინა* მოდელირების დროის 46%. იმისთვის, რომ რიგში დარეგისტრირებულიყო 1000 მოთხოვნა, საჭირო შეიქმნა 1027,915 წთ (*მოდელირების დრო*). ეს კონკრეტული მონაცემები მიღებულია შემთხვევითი რიცხვების გენერატორის გამოყენებით. თუ გამოვიყენებდით გენერატორის სხვა ნაკადებს ან სხვა გენერატორს, მაშინ რიცხვითი შედეგებიც სხვა იქნებოდა და ისინი მხოლოდ თეორიული $d(n)$, $q(n)$ და $u(n)$ მოსალოდნელი მნიშვნელობების შეფასებებია; თუმცა კი ეს პარამეტრები (გამოსავალი მონაცემები) შეადგენს შესავალი პარამეტრების ფუნქციას. მაშასადამე სანდო შეფასებების მისაღებად საჭიროა სტატისტიკის გამოყენება. ამისთვის კი უნდა შეიქმნას შეფასებათა შერჩევები:

$$\begin{array}{llll} \hat{d}^{(1)}(n), & \hat{d}^{(2)}(n), & \ldots, & \hat{d}^{(N)}N(n) \\ \hat{q}^{(1)}(n), & \hat{q}^{(2)}(n), & \ldots, & \hat{q}^{(N)}(n) \\ \hat{u}^{(1)}(n), & \hat{u}^{(2)}(n), & \ldots, & \hat{u}^{(N)}(n) \end{array}, \qquad (4.12)$$

რომლის ზედა ინდექსი მიუთითებს სიმულაციური მოდელირების *გატარების ნომერს.* ყოველი სიმულაციური გატარებისას უნდა გამოვიყენოთ *შემთხვევით რიცხვთა ახალი ნაკადი*!

სტატისტიკის კურსიდან ჩვენთვის ცნობილია (4.12) ტიპის შერჩევების დამუშავების სტატისტიკური მეთოდები (წერტილოვანი შეფასება, ნდობის ინტერვალები, სტატისტიკური ჰიპოთეზები, რეგრესიული ანალიზი და სხვ.), რომელთა გამოყენება აუცილებელია კვალიფიციური შეფასებების შესაქმნელად მასობრივი მომსახურეობის სისტემის მუშაობის შეფასებაზე, ასევე სხვა მოდელირებადი რთული სისტემებისთვის. აქვე შევნიშნოთ, რომ შეფასების სიზუსტე ასევე დამოკიდებულია გაჩერების პირობაზე – მოთხოვნათა დაკავების რიგზე (აქ $=1000$). მისი განსაზღვრის მეთოდები წარმოდგენილია [1]-ში.

ლისტინგი 4.10 (გამოსავალი მონაცემების ანგარიში)

მოთხოვნის დადგომებს შორის საშუალო დრო	1,000 წთ
კვანძის მიერ მომსახურეობის საშუალო დრო	0,500 წთ
მოთხოვნათა რიცხვი	1000
რიგში მოთხოვნის საშუალო დაკავება	0,430 წთ
რიგში მოთხოვნათა საშუალო რიცხვი (წთ-ში)	0,418
მომსახურეობის კვანძის დაკავებულობის კოეფიციენტი	0,460
მოდელირების პერიოდის ხანგრძლივობა	1027,915 წთ

მმსმეკ-ის M/M/1 მოდელისთვის მისი სიმარტივის გამო მოხერხდა (Ross, 1997) თეორიულად გამოთვლილი ყოფილიყო სისტემის 3 გამოსავალი პარამეტრი (შესაბამისი საწყისი შემავალი ინფორმაციით), რომლებიც ყველა 0,5-ის ტოლია. ცხადია, რომ მოდელირებით მიღებული შედეგები დამაკმაყოფილებელია. n -ის გაზრდით და მოდელირების გატარებათა რიცხვის შექმნით $(N \geq 30)$ შეფასებები საგრძნობლად დაზუსტდება.

შევნიშნოთ, რომ სხვადასხვა მოდელებისთვის იცვლება როგორც სისტემის მუშაობის შეფასების კრიტერიუმები, ასევე მოდელირების გაჩერების წესები, თუმცა მათი დაპროგრამების ლოგიკა ანალოგიურია და აქ წარმოდგენილი მასალის გაცნობის შემდეგ ნოვაციებზე გადასვლა გაიოლებული იქნება.

თავი 5. სიმულაციური მოდელების შექმნის ტექნოლოგიები

§5.1. სიმულაციური მოდელების შექმნის ალტერნატიული მიდგომები

სიმულაციური მოდელების შემქმნელები დღეს-დღეობით ქმნიან კომერციულ პროგრამულ უზრუნველყოფებს, სადაც იყენებენ სისტემების მოდელირების ყველაზე სრულყოფილ საშუალებებს და არსებულ თანამედროვე კომპიუტერის არტიტექტურასა და პროგრამულ უზრუნველყოფას.

§5.1.1. პარალელური და განაწილებული მოდელირება

მოდელი-მმსმეკ და მისი სიმულაციური მოდელირების პროგრამული უზრუნველყოფა და მისი მოქმედების ლოგიკა ისეა აგებული, რომ პარამეტრების ერთი გამოთვლები საწინდარია მეორე გამოთვლებისთვის. თანაც პროგრამას ასრულებს ერთი კომპიუტერი. ასეთ შემთხვევაში მოდელირებას *უწოდებენ მიმდევრობითს.*

ბოლო წლებში კომპიუტერული ტექნოლოგია საშუალებას იძლევა რამოდენიმე კომპიუტერის ერთ ლოკალურ ქსელში ჩართვას და *პარალელური გამოთვლების* ჩატარებას. ეს ეხება დაპარალებულ პროცესორებსაც, როცა პროგრამის ნაწილები *განაწილებულია ოპტიმალურად პროცესორებზე* შესრულების დროისა და სხვა მანქანური რესურსების დაზოგვის მიზნით.

ეს მიდგომა ასევე ვითარდება დინამიკურ მოდელირებაში. ამ მიმართულებით სიახლეები შეიძლება ნახოთ მაღალი რანგის საერთაშორისო მნიშვნელობის ჟურნალში: "ACM Transactions on Modeling and Computer Simulation" (TOMACS), ასევე ყოველწლიურ გამოცემაში: "Proceedings of the Winter Simulation Conference".

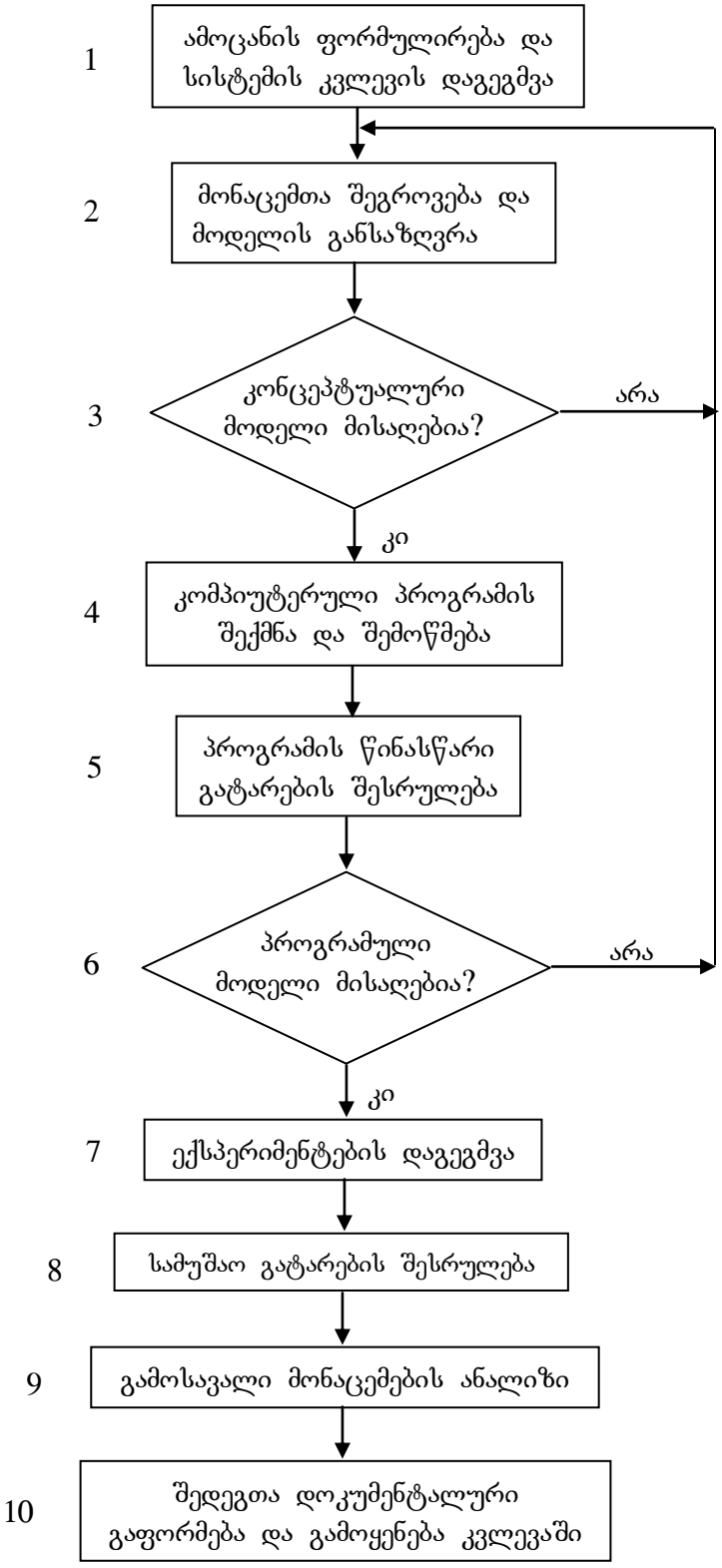
არსებობს მოდელირებადი დინამიკური სისტემის დაყოფის ბევრი საშუალება, ანუ მისი შესრულება განაწილდეს პარალელურად მოქმედი სხვადასხვა პროცესორებს შორის. ყველაზე ბუნებრივი განაწილებაა-განაწილება სხვადასხვა ფუნქციების მიხედვით. მაგალითად ზემოთ ჩამოყალიბებული ფუნქციები შესრულდეს სხვადასხვა პროცესორებზე. ძირითადი პროცესორი შეასრულებს იმ სამუშაოს სხვა პროცესორებთან პარალელურად, რაც ძირითად პროგრამას უნდა შეესრულებინა.

ასევე, სიმულაციური მოდელი შეიძლება ჩატარდეს რამოდენიმე პარალელურ კომპიუტერზე ქსელის გამოყენებით. აქ მდს-ს ნაცვლად მუშაობს პროცესორებს შორის *შეტყობინების გადაცემის* სისტემა, როდესაც ყოველ შეტყობინებას აქვს "*დროის ჭდე*".

ინტერნეტისა და www-ს გარემოში სიმულაციური მოდელირებისას ფართოდ გამოიყენება კლიენტ-სირვერის მექანიზმი.

§5.1.2. მოდელირებით სისტემების კვლევის ეტაპები

თუ გადავხედავთ გავლილ მასალას, დავრწმუნდებით, რომ დაპროგრამება ერთ-ერთი მნიშვნელოვანი ეტაპია რთული სისტემების მოდელირებისა და მისი შემდგომი ანალიზისთვის. ამასთან ერთად დიდი ყურადღება უნდა მიექცეს სხვადასხვა საკითხს: მოდელირების *გამოსავალი მონაცემების სტატისტიკურ ანალიზსა* და *პროექტის მენეჯმენტს.* ნახ. 5.1-ზე ნაჩვენებია ეტაპები, რომელიც შეადგენს *მოდელირებით სისტემების კვლევის* მიმდინარეობას (Law, 1999) [1]. ეტაპის ნომერი სქემიდან გაშლილია ქვემოთ უფრო დაწვრილებით. შევნიშნოთ, რომ რთული სისტემის კვლევა მოდელირებით მარტივი მიმდევრობითი მეთოდი არაა და ხშირად გვიხდება უკანა ეტაპზეც დაბრუნება.



ნახ. 5.1. მოდელირებით სისტემის კვლევის ეტაპები

1. ამოცანის ფორმულირება და სისტემის კვლევის დაგეგმვა

1.1. ხელმძღვანელის მიერ ამოცანის დასმა.

1.2. სისტემის შესწავლის მიზნით მთელი რიგი შეხვედრების ჩატარება პროექტის ხელმძღვანელსა, სიმულაციური მოდელირების ანალიტიკოსებსა და შესასწავლი თემატიკის ექსპერტებს შორის, სადაც უნდა განიხილებოდეს შემდეგი საკითხები:
   - კვლევის გლობალური მიზანი;
   - სპეციალური საკითხები, რომელზეც უნდა უპასუხოს კვლევამ;
   - სამუშაოს ხარისხის კრიტერიუმები, გამოყენებული მომავალში სისტემის სხვადასხვა კონფიგურაციების ეფექტურობის შესაფასებლად;
   - მოდელის მასშტაბი;
   - სისტემის მოდელირებადი კონფიგურაციები;
   - გამოყენებული პროგრამული უზრუნველყოფა;
   - კვლევისთვის გამოყოფილი დროის ინტერვალი და აუცილებელი რესურსები.

2. მონაცემთა შეგროვება და მოდელის განსაზღვრა

2.1. სისტემის კონფიგურაციაზე და ექსპლოატაციის საშუალებებზე ინფორმაციის შეგროვება, რომელსაც მივყავართ განმარტებაზე:
   - მოცემულ შემთხვევაში ერთი სპეციალისტი არაა საკმარისი;
   - შესასწავლი თემატიკის მიხედვით საჭიროა ექსპერტების ჩართვა, რომელთაც გააჩნიათ ხარისხიანი და სარწმუნო ცოდნა და ინფორმაცია;
   - ექსპლოატაციის საშუალებების ფორმალიზება არ შეიძლება;

2.2. მონაცემთა შეგროვება (თუ ეს შესაძლებელია) მოდელის პარამეტრთა და ალბათობათა შესავალი განაწილების შეფასებისთვის.

2.3. კონცეპტუალური მოდელის მონაცემთა სქემატური წარმოდგენა.

2.4. მონაცემთა შეგროვება არსებული მოდელის სამუშაო მახასიათებლებზე.

2.5. მოდელის დეტალურობის დონის არჩევა ქვემოთ ჩამოთვლილი ფაქტორების მიხედვით:
   - პროექტის მიზანი;
   - მუშაობის ხარისხის კრიტერიუმები;
   - მონაცემთა წვდომა;
   - მნიშვნელოვნების ინტერესები;
   - კომპიუტერული შესაძლებლობები;
   - შესასწავლ თემატიკაზე ექსპერტთა აზრი;
   - შეზღუდვები, დაკავშირებული დროსთან და ფინანსებთან;

2.6. მოდელის ყოველ ელემენტსა და სისტემიდან მისი შესაბამის ელემენტს შორის ურთიერთცალსახა დამოკიდებულების დადგენა.

2.7. რეგულარული ერთიერთქმედების ჩამოყალიბება პროექტის ხელმძღვანელსა და პროექტის სხვა წამყვან სპეციალისტებს შორის.

3. კონცეპტუალური მოდელის ადექვატურობის განსაზღვრა

3.1. კონცეპტუალური მოდელის სტრუქტურული ანალიზის შესრულება იმ დაშვებების აღწერს წარმოდგენით, რომელიც განსახილველად გადაეცემა აუდიტორიას. აუდიტორია შედგება ხელმძღვანელის, ანალიტიკოსებისა და თემატიკის შემსწავლელი ექსპერტებისგან. ყურადსაღებია ის, რომ:
   - ანალიზი გვეხმარება დავრწმუნდეთ, რომ დაშვებები, რომელიც მისაღებია მოდელისთვის, სწორია და არაფერია გამოტოვებული;
   - ანალიზი უზრუნველყოფს საკუთრების უფლებას მოდელზე;
   - ანალიზი სრულდება დაპროგრამების წინ, რათა არ დავუშვათ მოდელის შესაძლო *გადაპროგრამება.*

4. კომპიუტერული პროგრამის შექმნა და მისი შემოწმება

4.1. მოდელის დაპროგრამება C-ენაზე ან სხვა ენაზე და მოდელირების პროგრამული უზრუნველყოფის გამოყენება (ნაწილობრივ, Arena, AutoMod, Extend, ProModee, WITNESS). დაპროგრამების ენების გამოყენების უპირატესობა პროგრამული უზრუნველყოფით სარგებლობასთან მდგომარეობს იმაში, რომ მას აქვს *შესყიდვის* დაბალი ფასი და ენაზე შექმნილი მოდელი როგორც ყოველთვის უფრო სწრაფად სრულდება. თუმცა პროგრამული უზრუნველყოფით სარგებლობა ამცირებს დაპროგრამებაზე დახარჯულ დროს და ჯამში გვაძლევს *მთელი პროექტის ნაკლებ ღირებულებას.*

4.2. მოდელირების კომპიუტერული პროგრამის შემოწმება და მისი გამართვა.

5. პროგრამის წინასწარი გატარებების შესრულება

5.1. წინასწარი გატარებების შესრულება მე-6 ეტაპზე პროგრამული მოდელის შემოწმების განხორციელების მიზნით.

6. პროგრამული მოდელის შესატყვისობის შემოწმება

6.1. თუ არის არსებული, რეალური სისტემა, მაშინ პროგრამული მოდელის მუშაობის ხარისხისა და არსებული სისტემის მუშაობის ხარისხის კრიტერიუმთა შედარება.

6.2. რეალური სისტემის არსებობის შემთხვევაში ანალიტიკოსებისა და ექსპერტების მიერ დამოუკიდებლად წინასწარი გატარებების დათვალიერება მისი მოქმედების სისწორის შეფასებისათვის.

6.3. *მგრძნობელობის ანალიზის* გამოყენება მოდელის ფაქტორების დასადგენად, რომელთაც გააჩნიათ მნიშვნელოვანი გავლენა სისტემის მუშაობის ხარისხზე, რამეთუ მათი მაღალი სიზუსტით მოდელირებაა საჭირო.

7. ექსპერიმენტის დაგეგმვა

7.1. დეტალური წარმოდგენა იმისა, თუ რა არის მნიშვნელოვანი ყოველი სისტემური კონფიგურაციისთვის:

- ყოველი გატარების ხანგრძლივობა;
- თუ აუცილებელია გადასასვლელი პერიოდის ხანგრძლივობის განსაზღვრა;
- სიმულაციური მოდელირების დამოუკიდებელი გატარებები განსხვავებული შემთხვევითი რიცხვების გამოყენებით, რაც აუცილებელია კვალიფიციური სტატისტიკური შეფასებების შესაქმნელად. მაგ., მაღალი ნდობით ინტერვალების აგება კრიტერიუმების პარამეტრებისთვის და ა. შ.

8. სამუშაო გატარებათა შესრულება

8.1. შედეგების მისაღებად, რომელიც აუცილებელია მე-9 ეტაპის შესასრულებლად, სამუშაო გატარებათა შესრულება

9. გამოსავალი მონაცემების ანალიზი

9.1. გამოსავალი მონაცემების დამუშავება შემდეგი მიზნით:

- სისტემის ცნობილი კონფიგურაციის აბსოლიტური მახასიათებლების განსაზღვრა-შეფასება.
- სისტემის ალტერნატიული კონფიგურაციების შედარება.

10. შედეგების დოკუმენტური წარმოდგენა და სარგებლობა

10.1. კომპიუტერული პროგრამით მიღებული შედეგების (მონაცემების) დოკუმენტური გაფორმება. მიმდინარე პროექტის გამოყენებისა და მომავალი პროექტის კონსტრუირების მიზნით მიღებული შედეგების გამოკვლევა.

10.2. კვლევის შედეგების წარმოდგენა:
- ხელმძღვანელობისთვის და მათთვის, რომლებიც მიეკუთვნებიან აუდიტორიას და არაა ცნობილი ყველა შედეგი. მოდელის ახსნისა და წარმოდგენისთვის სხვადასხვა საშუალებების, განსაკუთრებით კი *ანიმაციის გამოყენება.*
- მოდელის შესაქმნელად, დანიშნულებისა და მნიშვნელობისთვის განხილვის პროცესის მოწყობა, რათა გაიზარდოს მოდელის სანდოობა მომხმარებელზე.

10.3. გადაწყვეტილების მიღების პროცესში მიღებული შედეგების გამოყენება.

§5.2. სიმულაციური მოდელირების უპირატესობა, ნაკლოვანებანი და შეცდომები

წარმოგიდგენთ მოდელირების უპირატესობათა და ნაკლოვანებების ჩამონათვალს. ასევე განვიხილავთ იმ ტიპიურ შეცდომებს, რომელიც ხშირია მოდელირებაში და რომელსაც შეუძლია დაარღვიოს მოდელირების პროცესი და შეიძლება პროექტი უსარგებლოდ გადააქციოს.

როგორც ითქვა, სიმულაციური მოდელირება ფართოდ გამოიყენება რთული სისტემების ანალიზში. რთული სისტემის მაგალითია მასობრივი მომსახურეობის სისტემა, ხოლო *მასობრივი მომსახურეობის სისტემის კონკრეტული მაგალითებია* (ნახ. 5.2):

სისტემა	მომსახურეობის კვანძი	მოთხოვნები
ანკი	სალაროები	კლიენტები
საავადმყოფო	ექიმები, სანიტრები, საავადმყოფოს საწოლები	პაციენტები
კომპიუტერი	ცენტრალური პროცესორი,	შეტანა-გამოტანის მოწყობილობა
ჭარმოება	დანადგარები, მოსამსახურე-მუშები	დეტალები, აუცილებელი პროდუქტის შესაქმნელად
აეროპორტი	ასაფრენ-დასაფრენი ბილიკები, მგზავრთა რეგისტრაციის პუნქტები	თვითმფრინავები, მგზავრები
კავშირის ქსელი	კვანძები, კავშირის ხაზები	შეტყობინებები, პაკეტები

ნახ. 5.2

სხვადასხვა სფეროში რთული სისტემების სიმულაციური მოდელირება პოპულუარულია მისი უპირატესობებით:

- სტოქასტური ელემენტების მქონე რთული რეალური სისტემების უმრავლესობა *ზუსტად არ შეიძლება აღიწეროს მათამატიკური მოდელებით*, რომლებიც შეიძლება შეფასდეს და გამოკვლეული იყოს ანალიზურად. ამიტომ სიმულაციური მოდელირება ხდება *ერთადერთ შესაძლო მეთოდად* ასეთი სისტემების კვლევისთვის.
- ექსპლოატაციის გარკვეულ საპროექტო პირობებში მოდელირება საშუალებას იძლევა შევაფასოთ არსებული სისტემის საექსპლოატაციო მახასიათებლები.

- მოდელირების გზით შესაძლებელია შევადაროთ სისტემის პროექტთა ალტერნატიული ვარიანტები (ან სისტემისთვის ექსპლოატაციის ალტერნატიული სტრატეგიები), რათა განისაზღვროს, რომელი მათგანი უფრო შეესაბამება წარმოდგენილ მოთხოვნებს.
- მოდელირებით ჩვენ შეგვიძლია უზრუნველყოთ ექსპერიმენტის პირობების გაცილებით ეფექტური კონტროლი.
- მოდელირება საშუალებას იძლევა შევისწავლოთ სისტემის (მაგალითად კომერციული, ეკონომიკური და სხვ.) ფუნქციონირების ხანგრძლივი ინტერვალი მოკლე დროში (ვადებში) ან პირიქით, დაწვრილებით შევისწავლოთ სისტემის მუშაობა დროის გაშლილ ინტერვალში.

რა თქმა უნდა მოდელირებას გააჩნია თავისი ნაკლოვანებები:

- სტოქასტური სიმულაციური მოდელის ყოველი გატარება საშუალებას გვაძლევს მივიღოთ მოდელის ნამდვილი მახასიათებლების *მხოლოდ შეფასებანი* შესავალი პარამეტრთა გარკვეული ერთობლიობისთვის. აქედან გამომდინარე შესავალი პარამეტრთა ყოველი ერთობლიობისთვის საჭირო იქნება მოდელის რამოდენიმე დამოუკიდებელი გატარება. ამ მიზეზით სიმულაციური მოდელები არც თუ მოსახერხებელია ოპტიმიზაციისთვის. ამიტომ თუ ხელმისაწვდომია ან ადვილად კონსტრუირებადია ადექვატური სისტემის *ანალიზური მოდელი*, უკეთესია ვისარგებლოთ ამით და არა სიმულაციური მოდელირებით.
- ჩვეულებრივ სიმულაციური მოდელების დამუშავება ძვირი ღირს და ბევრ დროს მოითხოვს.
- მონაცემთა დიდ რაოდენობას, რომელიც სიმულაციური მოდელირებიდან მიიღება, მივყავართ იქამდე, რომ ისინი მკვლევარებში უფრო დიდ ნდობას მოიპოვებენ, ვიდრე ამართლებენ რეალობაში.
- თუ სიმულაციური მოდელი არაა შესწავლილი სისტემის ადექვატურობაზე, მაშინ მოდელირების შედეგები შეიცავს ნაკლებად სასარგებლო ინფორმაციას რეალურ სიტემაზე.

როდესაც ვღებულობთ გადაწყვეტილებას გვაწყობს თუ არა სიმულაციური მოდელირების გამოყენება სისტემის კვლევისთვის, უნდა გავითვალისწინოთ ყველა უპირატესობა და ნაკლოვანება, რომელიც შეიძლება თან ახლდეს მოდელირებას.

მივაქციოთ ყურადღება იმას, რომ ზოგიერთი სისტემის კვლევისას შესაძლებელია ვისარგებლოთ როგორც ანალიზური მოდელით, ასევე სიმულაციური მოდელირებით. კერძოდ, მოდელირება შეიძლება გამოვიყენოთ იმ თეორიული დაშვებათა დაფუძნებისათვის (შემოწმების გზით), რომელიც აუცილებელია ანალიზური მოდელის ფუნქციონირებისთვის.

დავუშვათ, რომ სისტემის კვლევისთვის მაინც არჩეულია მოდელირება. ჩამოვაყალიბოთ ყველა ის გარემოებანი, რომლებმაც შეიძლება ხელი შეუშალოს სიმულაციური მოდელირებით კვლევის წარმატებული შედეგების მიღებას (Law and McComs, 1998):

- მოდელირებით სისტემის კვლევის დასაწყისში ამოცანის არამკაფიო დასმა;
- მოდელის დეტალების დამუშავების არასაკმარისი დონე;
- მოდელირებით სისტემის კვლევის პროცესში ხელმძღვანელთა შეთანხმებულობის არასაკმარისობა;
- ხელმძღვანელობის მიერ მოდელირების არასწორი გაგება;
- მოდელირებით სისტემის კვლევის მიდგომა, როგორც დაპროგრამებაში მარტივი სავარჯიშოს გადაწყვეტა;
- გუნდში, რომელიც მუშაობს მოდელირებაზე, იმ სპეციალისტების არყოფნა ან ნაკლებობა, რომლებმაც იციან სიმულაციური მოდელირების მეთოდოლოგია;
- სისტემაზე არასაკმარისი მონაცემების მოგროვება;
- მოდელირების პროცესში შეუსაბამო პროგრამული უზრუნველყოფით სარგებლობა;

- მოდელირებისთვის პროგრამული უზრუნველყოფით სარგებლობა ცხადი სახით. მაგალითად, რთული მაკროოპერატორები, რომელთა სარგებლობა სარწმუნოდ ვერ იქნება დოკუმენტალურად დადასტურებული, რადგან ისინი ვერ შექმნიან საჭირო მოდელურ ლოგიკას;
- ანიმაციის არასწორი სარგებლობა;
- რეალურ სისტემაში შემთხვევითი რიცხვების წყაროების არასწორი განმარტება;
- მოდელირებაში შემავალი შემთხვევითი პარამეტრების ნებისმიერი განაწილებით სარგებლობა;
- სიმულაციური მოდელის *მხოლოდ ერთი გატარებით* მიღებული გამოსავალი პარამეტრების სტატისტიკური ანალიზი;
- ალტერნატიული პროექტების შედარება მოდელირების ერთი გატარებით მიღებული მონაცემების ხარჯზე;
- მოდელის მუშაობის შეფასების არასწორი კრიტერიუმებით სარგებლობა;

§5.3. კომპონენტები და აღნიშვნები მასობრივი მომსახურეობის სისტემებში

მასობრივი მომსახურეობის სისტემა შეიცავს 3 ძირითად კომპონენტს: *მოთხოვნის დადგომის პროცესი, მომსახურეობის მექანიზმი და მომსახურეობის დისციპლინა.*

დადგომის პროცესი შედგება მასობრივი მომსახურეობის სისტემაში მოთხოვნის გამოჩენის მექანიზმის აღწერისგან, თუ A_i - $(i-1)$ და (i) მოთხოვნათა დადგომებს შორის დროა, მაშინ $A_1, A_2, \ldots$ დამოუკიდებელი, ერთნაირად განაწილებული შემთხვევითი სიდიდეთა მიმდევრობაა, და დადგომებს შორის *მოსალოდნელ საშუალო მნიშვნელობას* $\beta = E(A)$ -თი აღვნიშნავთ, მაშინ $\lambda = \frac{1}{\beta} = \frac{1}{E(A)}$ იქნება *მოთხოვნათა დადგომის ინტენსივობა.*

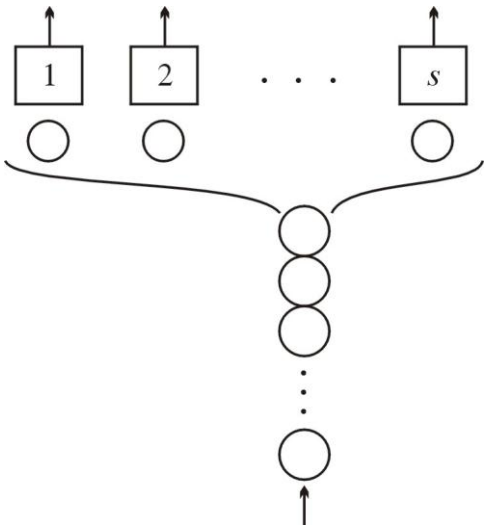
მომსახურეობის მექანიზმი განისაზღვრება შემდეგი ფაქტორებით: მომსახურეობის კვანძების რაოდენობა (s) ; თითოეული კვანძისთვის ცალკე რიგის ან საერთო რიგის არსებობა და კვანძის მიერ მომსახურეობაზე დახარჯული დროის ალბათური განაწილება. ვთქვათ i -ურ დამდგარ მოთხოვნაზე S_i -მომსახურეობის დროა, მაშინ $S_1, S_2, \ldots$ -დამოუკიდებელი და ერთნაირად განაწილებული შემთხვევითი სიდიდეების მიმდევრობაა, რომლის მომსახურეობის მოსალოდნელი საშუალო დროა- $E(S)$. *მომსახურეობის სიჩქარე გაინსაზღვრება* $\omega = 1/E(S)$ -ფორმულით.

მომსახურეობის დისციპლინა განისაზღვრება წესით, რომელსაც მომსახურეობის კვანძი იყენებს რიგიდან შემდეგი მოთხოვნის ასარჩევად. გამოიყენება რიგის შემდეგი დისციპლინები:

- FIFO (First-In, First-Out): "პირველი მოხვედი – პირველს მოგემსახურებიან" პრინციპი;
- LIFO (Last-In, First-Out): "ბოლო მოხვედი – პირველს მოგემსახურებიან" პრინციპით;
- *პრიორიტეტი*: მოთხოვნაზე მომსახურეობის რიგის დაწესება მათზე მოთხოვნის, საჭიროების და მიხედვით.

თუ სისტემაში დანერგილია მომსახურეობის FIFO პრინციპი და სისტემა შედგება მომსახურეობის s კვანძისგან ზემოთ ჩამოთვლილი კომპონენტებით. ასეთი სისტემა *GI/G/s*-სიმბოლოთი აღინიშნება, სადაც *GI* (General Independent-ნებისმიერი დამოუკიდებელი) ეხება $\{A_i\}$ სიდიდეების განაწილებას, *G* (General-ნებისმიერი) ეხება $\{S_i\}$ სიდიდეთა განაწილებას, s -მომსახურეობის კვანძების რაოდენობაა. თუ A_i და S_i სიდიდეები მოიცემა სპეციალური განაწილებებით (ეს ასეა ყოველი კონკრეტული მოდელირებისას), მაშინ *GI* და *G* სიმბოლოების ნაცვლად გამოიყენება ამ განაწილების აღნიშვნები. *ექსპონენციალური* განა-

წილების შემთხვევაში გამოიყენება სიმბოლო M. სიმბოლო E_k გამოიყენება *ერლანგის* k-ური რიგის განაწილებისთვის; D-სიმბოლო გამოიყენება დეტერმინისტული ან მუდმივი დროის ინტერვალებისთვის. მმსმეკ-სთვის FIFO დისციპლინით გამოიყენება აღნიშვნა – $M/M/1$.



ნახ. 5.3. GI/G/s – სისტემა

$GI/G/s$-ნებისმიერი სისტემისთვის $\rho = \lambda/(s\omega)$ სიდიდეს *გამოყენების კოეფიციენტი* ეწოდება. $s\omega$-სისტემაში მომსახურეობის სიჩქარეა, როდესაც მომსახურეობის ყველა კვანძია დაკავებული. ρ მიუთითებს იმაზე, თუ რამდენადაა სისტემის რესურსები გამოყენებული მოდელირებაში.

§5.4.* მასობრივი მომსახურეობის სისტემის მუშაობის შეფასების დამატებითი კრიტერიუმების შესახებ

არსებობს სისტემის მუშაობის შეფასების კრიტერიუმების სიმრავლე. ამ პარაგრაფში წარმოდგენილია მხოლოდ ის ოთხი მახასიათებელი, რომელიც ჩვეულებრივ გამოიყენება მასობრივი მომსახურეობის სისტემების კვლევისას.

ვთქვათ D_i – i-ური მოთხოვნის რიგში დაკავების დროა;

$W_i = D_i + S_i$ – i-ური მოთხოვნის სისტემაში ყოფნის დროა;

$Q(t)$ – დროის t მომენტისთვის რიგში მოთხოვნათა რიცხვია;

$L(t)$ – დროის t მომენტისათვის სისტემაში მოთხოვნათა რიცხვია ($Q(t)$-ს ემატება მოთხოვნათა რიცხვი, რომელიც მომსახურეობაზეა t მომენტში);

მაშინ მახასიათებლებს

$$d = \lim_{n\to\infty}\left(\sum_{i=1}^{n}\Delta_{i\,\text{ც}}\right)\Big/ n \;-\; \text{ალბათობით } 1;$$

და

$$w = \lim_{n\to\infty}\left(\sum_{i=1}^{n}W_{i\,\text{ც}}\right)\Big/ n \;-\; \text{ალბათობით } 1,$$

თუ ეს ზღვრები არსებობს, შესაბამისად ეწოდებათ *საშუალო დაგვიანება* და *მოლოდინის საშუალო დრო.*

მახასიათებლები

$$Q = \lim_{T\to\infty}\left(\int_0^T Q(t)dt\right)\Big/T \quad - \text{ ალბათობით } 1;$$

და

$$L = \lim_{T\to\infty}\left(\int_0^T L(t)dt\right)\Big/T \quad - \text{ ალბათობით } 1,$$

თუ ეს ზღვრები არსებობს, შესაბამისად *რიგში მოთხოვნათა რიცხვია* გათვლილი ერთეულოვან დროზე და *სისტემაში მოთხოვნათა რიცხვია* გათვლილი ერთეულოვან დროზე. განმარტებებში "ალბათობით 1" ცნება უფრო მათემატიკური სიზუსტისთვის წერია და მას პრაქტიკული რეალიზაციაში არავითარი მნიშვნელობა არ აქვს. შევნიშნოთ, რომ $\rho = \lambda/(s\omega) < 1$ აუცილებელი პირობაა d , w , Q და L პარამეტრების არსებობისთვის $GI/G/S$ -სისტემაში. პარამეტრებს შორის არსებობს კავშირი *შენახვის განტოლების* სახელწოდებით:

$$Q = \lambda d \qquad \text{და} \qquad L = \lambda W .$$

არსებობს კიდევ ერთი მნიშვნელოვანი განტოლება

$$W = d + E(S).$$

$M/M/1$ სისტემისთვის მტკიცდება, რომ $L = \rho/(1-\rho)$.

§5.5. სიმულაციური მოდელირების პროგრამული უზრუნველყოფის შესახებ

ამ თავში წარმოდგენილი სისტემების მოდელირებისას გავარკვიეთ, რომ უმრავლესობა დისკრეტულ-ხდომილებითი მოდელებისთვის, აუცილებელია პროგრამული უზრუნველყოფის შექმნა. ასეთი მოდელების საერთო ფუნქციონალური შესაძლებლობათა დადგენის მიზნით დაპროგრამებაში მივიდნენ იქამდე, რომ დამუშავებულიყო სიმულაციური მოდელირების სპეციალური პროგრამული *პაკეტები*. უფრო მეტიც, ასეთი პაკეტებით სარგებლობის სიმარტივემ და ეფექტურობამ დიდი როლი ითამაშა სიმულაციური მოდელირების პოპულარობის ზრდაზე.

მოკლედ შევეხოთ სიმულაციური მოდელირების პროგრამების კლასიფიკაციას. ჯერ მათი გარკვეული უპირატესობათა შესახებ.

- უმრავლესობა სიმულაციური მოდელირების პაკეტებისა ავტომატურად ფლობენ ფუნქციონალურ შესაძლებლობებს, რათა აიგოს მოდელი, რაც არსებითად ამცირებს დაპროგრამებისთვის გამოყოფილ დროს და პროექტის საერთო ღირებულებას;
- სიმულაციური მოდელები, რომლებიც შექმნილია მოდელირების პაკეტებით, როგორც წესი იოლად მოდიფიცირდება და გამოიყენება;
- სიმულაციური მოდელირების პაკეტები უზრუნველყოფენ სიმულაციური მოდელის შექმნისათვის ბუნებრივ გარემოს;
- სიმულაციური მოდელირების პაკეტები უზრუნველყოფენ მოდელში შეცდომების აღმოჩენის უფრო სრულყოფილ მექანიზმს, რამეთუ ისინი ასრულებენ მრავალი ტიპის შეცდომების ავტომატურ ძიებას. ამით მცირდება სერიოზული შეცდომების დაშვების შესაძლებლობა.

მიუხედავად ზემოთ ჩამოთვლილი უპირატესობებისა, ზოგიერთი სიმულაციური მოდელი (ეს განსაკუთრებით ეხება სამხედრო-თავდაცვით სფეროს) კვლავ იქმნება მოდელირების უნივერსალურ ენაზე, რომელსაც აქვს გარკვეული უპირატესობანი:

- უმრავლესობა მოდელების შემქმნელებისა ფლობს დაპროგრამების ენებს, რასაც ვერ ვიტყვით სიმულაციური მოდელირების სპეციალურ პროგრამულ პაკეტებზე;

- *C* ან *C++* ენებზე დაწერილი სიმულაციური მოდელის გატარება უფრო სწრაფია, ვიდრე იმ მოდელებისა, რომლებიც შექმნილია პაკეტებით;
- უნივერსალურ ენაზე შექმნილი მოდელები გამოირჩევიან დიდი მოქნილობით, ვიდრე პაკეტები;
- უნივერსალურ ენაზე შექმნილი პროგრამული უზრუნველყოფა გაცილება იაფია (თუმცა პროექტის საერთო ღირებულება ყოველთვის არა);
- ხშირად, როდესაც მოდელირება იქმნება პროგრამული პაკეტებით, აუცილებელი ხდება პროგრამის გარკვეული მოდულები შეიქმნას უნივერსალურ ენაზე.

არსებობს სიმულაციური მოდელის შექმნის პროგრამული პაკეტების ორი ტიპი: *უნივერსალური პაკეტები* და *საგანზე-ორიენტირებული* პაკეტები. უნივერსალური პაკეტები გამოიყენება გარკვეული ჯგუფი მოდელების ასაგებად, ხოლო საგანზე ორიენტირებული კი *სპეციალური* ამოცანების გადასაწყვეტად მოდელირებაში.

ჩამოვთვალოთ სიმულაციური მოდელირების პროგრამული უზრუნველყოფის ძირითადი მახასიათებლები: *მოქნილობა მოდელირებაში, სიმარტივე გამოყენებაში; მოდელის მოქმედების მაღალი სიჩქარე;* მოსახერხებელი "მეგობრული" *ინტერფეისი*; *სამუშაო ვერსიის შექმნა, რათა გაკეთდეს მონაცემთა ექსპორტი და იმპორტი სხვა* სისტემებიდან; *გააკეთოს მოდელირება სხვადასხვა სცენარებით*, რომლებიც მეორდება მოდელირების პარამეტრების ცვლილებებით; *სისტემის ინიციალიზაცია* მისი დაკავებულობის მდგომარეობაში; მოდელის გატარების ბოლოში *მოდელის მდგომარეობის შენახვა*; *სიმულაციური მოდელის ღირებულება* (დღევანდელი დღისთვის სიმულაციური მოდელირების პროგრამულ უზრუნველყოფათა ღირებულება იცვლება 500-დან 50000 ამერიკულ დოლარამდე. აქ არ შედის სხვადასხვა ხარჯები, როგორიცაა დანახარჯები ექსპლოატაციაზე და მომსახურეობაზე, მოდერნიზაციაზე და ა.შ.).

სიმულაციური მოდელირების პაკეტების თითქმის უმეტესობა პერსონალურ კომპიუტერებზე მუშაობენ Windows სისტემით. თუ პაკეტები მუშაობენ *სხვადასხვა პლატფორმებზე*, მაშინ მათ უნდა უზრუნველყოფენ *ერთმანეთთან თავსებადობა*. ასევე უნდა გვახსოვდეს ოპერატიული მეხსიერების აუცილებელი მოცულობა და ის ოპერაციული სისტემა, რომელიც უზრუნველყოფს პაკეტის მუშაობას.

სიმულაციური მოდელირების პაკეტებისადმი მოთხოვნების ზრდა იმითაცაა განპირობებული, რომ სიმულაციური პროცესის ვიზუალიზაცია წარმოიდგინება *ანიმაციისა* და *გრაფიკის* მაღალი ხარისხით და შესრულებით. მნიშვნელოვანია თანამედროვე სტატისტიკური შესაძლებლობებიც, რომელსაც ფლობს პაკეტები.

პროგრამული უზრუნველყოფის შემქმნელები ყოველთვის მზად არიან მიაწოდონ ინფორმაცია *მომხმარებლებს* მათი პროდუქციის შესახებ, პარალელურად პროგრამის *უფასო ტესტირების ჩატარებით*! კარგი იქნება თუ შეიქმნება "*ინფორმაციული ბიულეტინი*" და ჩატარდება *კონფერენციები* პაკეტების მომხმარებლებისა და შემქმნელების მონაწილებით. პროგრამული პროდუქციის მწარმოებლები ასევე უნდა ზრუნავდნენ მათი პროგრამული უზრუნველყოფის შესაძლო *მოდერნიზაციაზე*.

სისტემის მუშაობის შესაფასებლად პროგრამაში ჩადებული უნდა იყოს *სტანდარტული ანგარიშების* შექმნის შესაძლებლობა, სადაც გამოსავალი მონაცემები უნდა წარმოდგნენ *მოცემულ ფორმატში*, მაგალითად როგორც ეს ხელმძღვანელობას სურს. ხშირად ანგარიშის წარმოდგენა აუცილებელი ხდება პროგრამის გატარების ნებისმიერ მოცემულ უბანზეც. ანგარიშებში ხშირად საჭიროა მაღალი ხარისხის გრაფიკის შექმნა სტატისტიკური მონაცემების წარმოდგენისთვის (ჰისტოგრამა, პოლიგონი, ფუნქციის დროზე დამოკიდებულება და ა. შ.).

და ბოლოს, აპრობირებული და მსოფლიოში აღიარებული უნივერსალური სიმულაციური სტოქასტური მოდელირების პროგრამულ პაკეტებზე: ძირითადად ორი პაკეტია: ARENA და EXTEND. ARENA შექმნილია Systems Modeling Corporation (1999, პენსილვანია) მიერ,

ხოლო Extend შექმნილია Imagine That, INC. (1997, კალიფორნია) მიერ. ამ პაკეტების და არა მარტო ამ პაკეტების შესახებ დაწვრილებით იხ. [1]-ში.

§5.6. ამოცანები და სავარჯიშოები

5.6.1. აღწერეთ თქვენი აზრით ყველაზე ეფექტური კვლევის საშუალება (ნახ. 1.2-ის მიხედვით) შემდეგი სისტემებისთვის. ახსენით თქვენი პასუხი:
ა) არსებული ფაბრიკის პატარა უბანი;
ბ) ტრანსპორტის ავტოსტრადა, რომელიც განიცდის სერიოზულ გადატვირთვებს;
გ) არსებულ საავადმყოფოში სასწრაფოს პუნქტი;
დ) პიცას ადგილზე მიტანის პროცედურა;
ე) სააგენტოსთვის, რომელიც დაკავებულია აეროპორტში ავტომობილების გაქირავებით, ქალაქგარეთ ავტობუსის მუშაობის მართვა;
ვ) რაიონში საომარი მოქმედებების კავშირის ქსელი.

5.6.2. დავუშვათ რომ გადაწყვეტილია ყველა იმ სისტემის სიმულაციური მოდელირებით შესწავლა, რომელიც ჩამოთვლილია წინა მაგალითში. გაარკვიეთ, როგორი უნდა იყოს მოდელირება: სტატისტიკური თუ დინამიკური, დეტერმინისტული თუ სტოქასტური, უწყვეტი თუ დისკრეტული?

5.6.3. განსაზღვრეთ t მომეტისთვის სისტემაში მოთხოვნათა საერთო რიცხვი $L(t)$ (რიგში მოთხოვნები პლუს მოთხოვნები მომსახურეობაზე, თუ კი ასეთი არსებობს) მმსმეკ-სთვის (წინა თავის §4.1-ში ჩამოყალიბებული სისტემისთვის).
ა) სამართლიანია თუ არა ტოლობა $L(t) = Q(t) + 1$. დააფუძნეთ თქვენი პასუხი.
ბ) სისტემის რეალიზაციისთვის, აღწერილი "ხელით" მოდელირებისას ააგეთ $L(t)$ ფუნქციის გრაფიკი (ანალოგიურად §4.2-ის წარმოდგენებისა) 0-სა და $T(6)$ დროის მომენტებს შორის.
გ) გრაფიკიდან, რომელიც მიიღება ბ) ამოცანის რეალიზაციის შედეგად, გამოთვალეთ $\hat{e}(6)$ - დროის ერთეულში მოთხოვნათა საშუალო რაოდენობა $[0, T(6)]$ ინტერვალის განმავლობაში.
დ) დაასრულეთ ნახ. 4.4, იმისთვის, რომ გავარკვიოთ თუ როგორ გამოითვლება $\hat{e}(6)$ მოდელირების პროცესში.

5.6.4. დავუშვათ, რომ მმსმეკ-ს კვლევისას (აღწერილი §4.1-ში), გადაწყვეტილია, რომ არ შეფასდეს რიგში მოსალოდნელი, საშუალო დაკავება. ამასთან სტრუქტურა და პარამეტრები უცვლელი რჩება. იცვლება თუ არა ამ დროს მდგომარეობის ცვლადები? თუ იცვლება, მაშინ როგორ?

5.6.5. განვიხილოთ მმსმეკ, რომელიც აღწერილია §4.1-ში. ვთქვათ, W_i – i-ური მოთხოვნის სისტემაში ყოფნის საერთო დროა, იგი შეადგენს რიგში ყოფნის დრო პლუს მისი მომსახურეობის დრო. სისტემის რეალიზაციისთვის, აღწერილი "ხელით" მოდელირებისას, წარმოდგენილი §4.2-ში, გამოთვალეთ $\hat{W}(m)$ – პირველი m მოთხოვნის სისტემაში ყოფნის საერთო დრო, სანამ ისინი არ დატოვებენ სისტემას. $m = 5$ -თვის შეავსეთ ნახ. 4.4. ამ შემთხვევაში როგორ შეიცვლება მდგომარეობის ცვლადები?

5.6.6. ნახ. 4.2-დან ჩანს, რომ რიგის მაქსიმალური სიგრძეა 3. ჩაწერეთ ამ პარამეტრის ზოგადი გამოსახულება (n–გაჩერების პარამეტრის გათვალისწინებით) და დაასრულეთ ნახ. 4.4, რათა რიგის სიგრძე ავტომატურად გამოითვლებოდეს მოდელირების პროცესში.

5.6.7.* შეცვალეთ მასობრივი მომსახურეობის სისტემის C-პროგრამული კოდი, ჩაწერილი §4.4-ში, რათა უზრუნველყოთ გამოთვლები და ისეთი დამატებითი მახასიათებლების ჩაწერა, როგორიცაა:

ა) სისტემაში მოთხოვნათა საშუალო რიცხვი გათვლილი დროის ერთეულზე (ამოცანა 5.6.3);

ბ) სისტემაში ხდომილების ყოფნის საერთო საშუალო დრო (ამოცანა 5.6.5);

გ) რიგში მაქსიმალური დაგვიანება;

დ) მაქსიმალური სიგრძის რიგი (ამოცანა 5.6.6);

ე) სისტემაში მოთხოვნის ყოფნის მაქსიმალური დრო;

ვ) მოთხოვნათა ნაწილი (წილი), რომელთა რიგში დაკავება აღემატება 1 წთ-ს.

ზ) ა)-ვ) ამოცანებისთვის შევქმნათ C -კოდი და გავუშვათ ეს პროგრამა, როცა გამოყენებულია შემთხვევითი რიცხვების იგივე გენერატორი.

შემავალი მონაცემები უცვლელია.

5.6.8.* §4.4-ში ჩაწერილი მასობრივი მომსახურეობის სისტემის C-პროგრამული კოდი გაატარეთ 10-ჯერ. დიდ ციკლში მოაქციეთ ძირითადი პროგრამის უპირატესი ნაწილი, დაწყებული ინიციალიზაციიდან დასრულებული ანგარიშების გენერატორის გამოძახებით (გამოიყენეთ შემთხვევითი რიცხვების გენერატორის ნაკადები). იმსჯელეთ შედეგებზე (ასეთ ქმედებას 10-ჯერადი დამოუკიდებელი *რეპლიკაცია* ეწოდება). შემავალი მონაცემები უცვლელია.

5.6.9.* წარმოვიდგინოთ, რომ მმსმეკ-ში, რომელიც აღწერილია §4.1-ში, მომსახურეობა იწყება დილის 9^{00} -ზე (დროის 0-ვანი მომენტი) და მთავრდება 17^{00} -ზე. მაგრამ მომსახურეობაზე მუშაობა სრულდება მაშინ, სანამ არ მოემსახურებიან უკანასკნელ შეკვეთას, რომელიც სისტემაშია 17^{00} -თვის. (რიგში ან მომსახურეობაზე). შეცვალეთ პროგრამის კოდი ისე, რომ აისახოს გაჩერების ეს პრინციპი და შეაფასეთ იგივე პარამეტრები. შემავალი მონაცემები უცვლელია.

5.6.10.** წარმოვიდგენოთ, რომ მმსმეკ-ში, რომელიც აღწერილია §4.2-ში, რიგში არის *მხოლოდ ორი ადგილი მოთხოვნის* დასაკავებლად და მოთხოვნა, რომელიც დადგება, როცა რიგი შევსებულია, *ტოვებს სისტემას* (ასეთ სიტუაციას ეწოდება *რიგიდან ამოვარდნა*). შექმენით ასეთი სისტემის სიმულაციური მოდელის *C-პროგრამული კოდი* მოდელირების გაჩერების ახალი წესით. მოდელირება მიმდინარეობს 480 წთ-ი. შეაფასეთ იგივე მაჩვენებლები, რომლეთა განმარტებები წარმოდგენილია §4.2-ში. ასევე ახალი პარამეტრი: მოსალოდნელი საშუალო *რიცხვი იმ მოთხოვნებისა, რომლებიც არ შეუერთნენ რიგს.* დანარჩენი შემავალი მონაცემები უცვლელია.

5.6.11.*** განვიხილოთ სისტემა s ($s \geq 1$) მომსახურეობის პარალელური კვანძით. დავუშვათ, რომ მოთხოვნების დადგომებს შორის დროის ინტერვალები დამოუკიდებელი ერთნაირად განაწილებული შემთხვევითი სიდიდეებია $E(A)$ -საშუალო მნიშვნელობით, ხოლო ერთი კვანძისგან დამოუკიდებლად მეორე კვანძის მიერ მომსახურეობაზე დახარჯული დროის ინტერვალები ასევე დამოუკიდებელი და ერთნაირად განაწილებული შემთვევითი სიდიდეებია $E(S)$ -საშუალოთი. თუ დგება მოთხოვნა და ერთი მაინც მომსახურეობის კვანძი თავისუფალია, მაშინ მისი მომსახურეობა დაიწყება მყისიერად, ამასთან თუ თავისუფალია რამოდენიმე კვანძი, მომსახურეობაში ჩაერთვება ყველაზე მარცხენა (დაბალი ნომრის მქონე) თავისუფალი კვანძი, წინააღმდეგ შემთხვევაში მოთხოვნა ჩადგება FIFO-ს პრინციპის საერთო ერთადერთი რიგის ბოლოში. ასეთი სისტემა აღინიშნება $M/M/s$ -სიმბოლოთი. დაწერეთ ამ სისტემის მოდელირების ზოგადი C-კოდი, რომელშიც შეფასდება ისეთი გამოსავალი პარამეტრები, როგორიცაა: რიგში მოსალოდნელი საშუალო დაკავება; რიგში მოსალოდნელი საშუალო რიცხვი

მოთხოვნებისა, გათვლილი ერთეულოვან დროზე; ყოველი კვანძის მიერ მომსახურეობაზე გამოყენების კოეფიციენტი (გაჩერების წესია n დაკავება რიგში). s, $E(A)$, $E(S)$ და n სიდიდეები მოდელირების შემავალი პარამეტრებია. ჩაატარეთ სიმულაციის გატარება, როცა

ა) $s=5$, $E(A)=1$, $E(S)=4$ და $n=1000$;

ბ) $s=6$, $E(A)=1/2$, $E(S)=3$ და $n=1000$;

გ) $s=10$, $E(A)=1/2$, $E(S)=6$ და $n=1000$.

5.6.12.* განმეორებით შეასრულე ამოცანა 5.6.11, მაგრამ ამ შემთხვევაში, როდესაც დადგება მოთხოვნა მომსახურეობაზე და რამოდენიმე კვანძი იქნება თავისუფალი, მაშინ მათ შორის არჩევანი გაკეთდება თანაბარი ალბათური განაწილებიდან. მაგალითად, თუ თავისუფალი იქნება 1, 3 და 5 კვანძები, მაშინ თითოეულის დაკავებულობა გათამაშდება 1/3 ალბათობით.

5.6.13.* კლიენტი შედის ბანკში, სადაც მუშაობს სამი პარალელური სალარო.

ა) თუ ბანკში მოქმედებს ერთი საერთო რიგი FIFO-ს პრინციპით, მაშინ როგორი უნდა იყოს ამ სისტემის სიმულაციური მოდელისთვის ხდომილებათა სია (ანუ სიის სიგრძე)?

ბ) თუ თითოეულ სალაროსთან წარმოიქმნება თავისი რიგი FIFO-ს პრინციპით, ამასთან კლიენტს შეუძლია ერთი რიგიდან მეორე რიგში *გადასვლა* (გადასვლით რიგის ბოლოში), მაშინ როგორი უნდა იყოს ამ სისტემის სიმულაციური მოდელირების ხდომილებათა სია?

5.6.14.* დღე-ღამე მომუშავე ბენზო-გასამართ სადგურს მომსახურეობის ერთ კვანძით (“კოლონკით”) ჰყავს *ორი ტიპის კლიენტი.* პოლიციის მანქანები შევსებაზე მიდიან ყოველი *30 წთ*-ის განმავლობაში. პირველი ასეთი მანქანა მიდის მუშაობის დაწყების დროის მომენტიდან *15 წთ*-ში. ჩვეულებრივი (არა პოლიციის) მანქანების სადგურაზე შევსებისთვის დადგომებს შორის დროის ინტერვალი ექსპონენციალური განაწილებისაა *5,6 წთ* საშუალო მნიშვნელობით. პირველი ჩვეულებრივი მანქანის კოლონკაზე დადგომის დროა *0 წთ.* ყოველი მანქანის მომსახურეობის დრო ექსპონენციალური განაწილებისაა *4,8 წთ.* საშუალო მნიშვნელობით. მანქანა, მისული კოლონკაზე, თუ ის თავისუფალია, მაშინვე დადგება მომსახურეობაზე. ჩვეულებრივი მანქანა, რომელიც მიდის შევსებაზე, დადგება რიგის ბოლოში, თუ კოლონკა დაკავებულია. პოლიციის მანქანა, რომელიც მიდის შევსებაზე და კოლონკა დაკავებულია, დგება რიგის თავში ნებისმიერი ჩვეულებრივი მანქანის წინ, ან თუნდაც წინა პოლიციის მანქანის წინ! თავიდან სისტემა თავისუფალია. სიმულაციური მოდელირება გრძელდება მანამ, სანამ რიგში არ დაკავდება ნებისმიერი ტიპის *500* მანქანა. დაწერეთ C - პროგრამული კოდი და შეაფასეთ რიგში დაკავების მოსალოდნელი საშუალო დრო თითოეული ტიპის მანქანისთვის; რიგში მოთხოვნათა საშუალო მნიშვნელობა და “კოლონკით” მოსალოდნელი სარგებლობის კოეფიციენტი.

თავი 6. ადექვატური და დეტალური სიმულაციური მოდელების შექმნა

§6.1. ზოგადი წარმოდგენები

ერთ-ერთი ყველაზე რთული პრობლემა, რომელსაც სიმულაციური მოდელირების ანალიტიკოსი აწყდება, მდგომარეობს განსაზღვრაში, არის თუ არა აგებული სიმულაციური მოდელი რეალური სისტემის ზუსტი წარმოდგენა? ანუ *ადექვატურია* თუ არა მოდელი რეალური სისტემისთვის?

დავიწყოთ ყველაზე მნიშვნელოვანი ტერმინების განსაზღვრით, ასეთებია: *მოდელის ვერიფიკაცია*, *ვალიდაცია* და *მოდელის სანდოობა.*

ვერიფიკაცია, ანუ შემოწმება *მოდელის დასავჯეროობაზე*, განისაზღვრება სწორად გარდაიქმნა თუ არა კონცეპტუალური სიმულაციური მოდელი კომპიუტერულ პროგრამაში, ანუ როგორ სრულდება მოდელირების კომპიუტერული პროგრამის გამართვა?! ცნება *ვერიფიკაცია* მარტივია, მაგრამ "დიდი" მოდელირების პროგრამის გამართვა საკმაოდ რთული ამოცანაა, რამეთუ მასში შეიძლება არსებობდეს მრავალი ლოგიკური გზა.

ვალიდაცია – ეს პროცესია, რომლითაც ვადგენთ, არის თუ არა სიმულაციური მოდელი (და არა კომპიუტერული პროგრამა!) რეალური სისტემის ზუსტი წარმოდგენა კვლევის კონკრეტული მიზნებისთვის.

სიმულაციური მოდელი და მისი რეზულტატები *დასაჯერებელია*, სარწმუნოა, სანდოა, თუ პროექტის მენეჯერი და სხვა ხელმძღვანელები აღიარებენ მათ სისწორეს, ჭეშმარიტებას და იყენებენ მას გადაწყვეტილების მიღებაში. შევნიშნოთ, რომ ადექვატური მოდელი არაა აუცილებელი იყოს დასაჯერებელი, ან პირიქით. მოდელი შეიძლება იყოს დასაჯერებელი, ამასთან არ გამოიყენონ გადაწყვეტილების მიღებისას. ქვემოთ წარმოდგენილია ის გარემოებანი, თუ როგორ დაადგინონ მოდელისადმი სანდოობა:

- მენეჯერის მიერ გაგებული უნდა იყოს ის დაშვებები, რომელიც მიღებულია მოდელში, და მასთან შეთანხმებული უნდა იყოს;
- მოდელის სისწორეზე და დასაჯერობაზე შემოწმების დემონსტრირება;
- უფლება საკუთრებაზე და მენეჯერის თანამონაწილეობა პროექტში;
- მოდელის შემქმნელების რეპუტაცია.

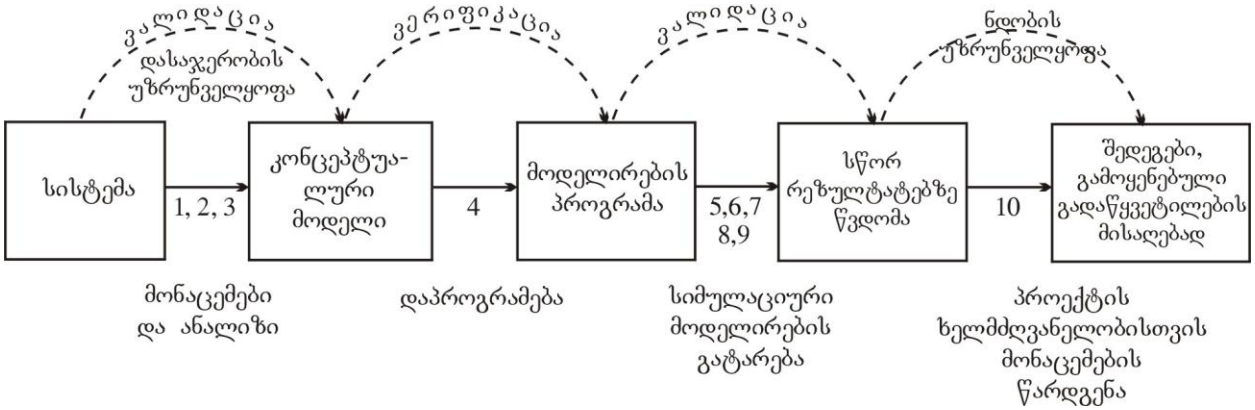
შეერთებული შტატების თავდაცვის სამინისტრო ხშირად სარგებლობს სიმულაციური მოდელირებით [1], სადაც ბოლო წლებში ინტერესს იჩენენ ვერიფიკაციასა, ვალიდაციასა და ისეთ კონცეფციაზე, როგორიცაა მოდელის *აკრედიტაცია* (VV&A). აკრედიტაცია – ოფიციალური აღიარებაა (შესაძლოა პროექტის სპონსორის მიერ იმ ფაქტისა, რომ სიმულაციური მოდელი ვარგისია განსაზღვრული მიზნებისთვის). ამოცანები, რომლებიც განიხილება აკრედიტაციის ჩატარებისას, გულისხმობს:

- მოდელის ვერიფიკაციისა და ვალიდაციის ჩატარებას;
- მოდელის შექმნისა და გამოყენების ისტორიას (ნაწილობრივ ვისი შექმნილია მოდელი და ა.შ.);
- ხელმისაწვდომ მონაცემთა ხარისხს;
- დოკუმენტაციის ხარისხს;
- სიმულაციურ მოდელთან დაკავშირებულ ცნობილ პრობლემებსა და შეზღუდვებს.

ნახ. 6.1-ზე ნაჩვენებია ვერიფიკაციის, ვალიდაციისა და მოდელისადმი სანდოობის დადგენის კოორდინაციისა და ურთიერთკავშირის სქემა. მართკუთხედი წარმოადგენს მოდელის მდგომარეობას. მუქი, ჰორიზონტალური ისრები შეესაბამება მოქმედებებს, აუცილებელს ერთი მდგომარეობიდან მეორეში გადასვლისთვის. ამოზნექილი, წყვეტილი ისრები აჩვენებს, სადაა

უკეთესი გამოვიყენოთ 3 მთავარი კონცეფცია (ვერიფიკაცია, ვალიდაცია, სანდოობა). ციფრები მუქი ისრების ქვეშ შეესაბამება მოდელირებით სისტემის შესწავლის ეტაპებს (იხ. §1.2.2).

ვალიდაციას შეიძლება დავუპირისპიროთ გამოსავალ მონაცემთა ანალიზი, რომელიც უფრო სტატისტიკური ამოცანაა. ამ დროს ჩვენ გვაინტერესებს სიმულაციური მოდელის გატარების ხანგრძლივობა, დამოუკიდებელ გატარებათა რიცხვი, რომელთათვისაც გამოიყენება განსხვავებული შემთხვევითი რიცხვები.



ნახ. 6.1

თუ გვინდა მივიღოთ სისტემის რაიმე მახასიათებლის საშუალოს *ოპტიმალური შეფასება*, ყურადღება უნდა მივაქციოთ როგორც მოდელის ვალიდაციას, ასევე გამოსავალი ელემენტის სტატისკურ ანალიზს.

§6.2. მოდელის დეტალიზაციის რეკომენდაციები

სიმულაციური მოდელირებისას პროექტის შემსრულებლებმა უნდა განსაზღვრონ, რთული სისტემის რა ასპექტებია საჭირო აისახოს სიმულაციურ მოდელში, დეტალიზაციის როგორი დონეა საჭირო მიღწეული იქნას; ასევე რა ასპექტები შეიძლება გამოვტოვოთ. რთული სისტემის ყველა ასპექტის მოდელირება ხშირად არ მოითხოვება და, როგორც ყოველთვის, ის მიუღწეველია დროსთან, ფინანსურ შეზღუდვებთან და კომპიუტერის შესაძლებლობებთან მიმართებაში.

გთავაზობთ რეკომენდაციებს, რათა განისაზღვროს სიმულაციური მოდელის დეტალიზაციის დონე (Law, 1991):

- კვლევითი ანალიზის დროს გულდასმით განსაზღვრეთ პრობლემები. აგებული მოდელები არ არსებობს უნივერსალურად ადექვატურნი, ისინი მუშავდება კონკრეტული მიზნებისთვის. თუ კვლევის პრობლემები არაა მოხაზული, შეუძლებელია გაინსაზღვროს მოდელის დეტალიზაციის საჭირო დონე. და ბოლოს, საჭიროა ზუსტად ვიცოდეთ, რა უნდა მენეჯერს პროექტისაგან. ძალიან გამართლებული მოდელის გამოყენებაც შეუძლებელია, თუ ის შექმნილი არაა საჭირო ამოცანის რეალიზაციისთვის;
- ყოველთვის საჭირო არაა დეტალური მოდელირება გავუკეთოთ სისტემის მნიშვნელოვან კომპონენტებს;
- სიმულაციური მოდელირების დამწყები სპეციალისტების ხშირი შეცდომა მდგომარეობს იმაში, რომ ისინი თავიანთ მოდელში ითვალისწინებენ ძალიან ბევრ დეტალს. მოდელის ადექვატურობის დასადგენად მოდელს წარუდგენენ სპეციალისტებს, ხელმძღვანელობასა და მენეჯერს საბოლოო დეტალიზაციის დონის განსაზღვრისთვის;

- პრაქტიკულად ყველა სიმულაციური კვლევა დაკავშირებულია დროისა და ფულადი სახსრების შეზღუდვებთან, რომელიც მთავარი ფაქტორია პროექტის დეტალიზაციაში.

§6.3. მოდელირების კომპიუტერული პროგრამების ვერიფიკაცია

გთავაზობთ მეთოდებს, რომლებიც შეიძლება გამოყენებული იყოს სიმულაციური მოდელირების კომპიუტერული პროგრამის გამართვისთვის.

- კომპიუტერული პროგრამა უმჯობესია დავწეროთ და გავმართოთ მოდულებად ან ქვეპროგრამებად, ფუნქციებად. მაგალითად: თუ სიმულაციური მოდელი შეიცავს 10000 ოპერატორს, ძნელი არაა მივხვდეთ, რომ აზრი არ აქვს დავწეროთ ერთიანი პროგრამა. თავიდან, როგორც ყოველთვის, წერენ და მართავენ ძირითად პროგრამას (ფუნქციას), აგრეთვე რამოდენიმე საკვანძო ქვეპროგრამასაც (ფუნქციასაც). შემდეგ თანდათანობით უნდა გაიმართოს დანარჩენი ქვეპროგრამებიც (ფუნქციებიც). ჯერ საკმარისია შესასწავლი რთული სისტემის *დეტალიზაციის საშუალო დონის* შექმნა, ვიდრე მაშინვე შევუდგეთ სრულ, *ოპტიმალურ დეტალიზაციას*. შემდეგ, მოთხოვნისდა მიხედვით *გავართულოთ დეტალიზაციის დონე* საჭირო ამოცანების რეალიზაციისთვის. მაგალითად: რამოდენიმე სალაროს მქონე ბანკის მოდელის აგებისას, როდესაც დასაშვებია კლიენტის ერთი რიგიდან მეორე რიგში გადასვლა, სასურველია დავწეროთ და შემდეგ გავმართოთ პროგრამა, რომელშიც კლიენტი არ გადადის რიგიდან რიგში. შემდეგი დეტალიზაცია იგულისხმებს უკვე გადასვლას;
- როდესაც მუშავდება მსხვილი სიმულაციური მოდელები, სასურველია, რომ კომპიუტერული პროგრამა გამართვაზე შეამოწმოს რამოდენიმე ადამიანმა, რამეთუ შემსრულებლები ველარ ამჩნევენ დაშვებულ შეცდომებს. ასეთ მიდგომას *სტრუქტურული გარჩევა* ეწოდება;
- იმისთვის, რათა დავრწმუნდეთ, რომ მიღებულია დამაკმაყოფილებელი შედეგები, საკმარისია სიმულაციური მოდელის გატარება სხვადასხვა შემავალი პარამეტრებისთვის;
- ერთ-ერთი ძლიერი მეთოდია *ტრასირება*. ამ დროს მოდელირებადი სისტემის მდგომარეობა, ანუ ხდომილებათა სიის შემცველობა, მდგომარეობის ცვლადები, ზოგიერთი სტატისტიკური მთვლელი და სხვა მონაცემები გამოიტანება მონიტორის ეკრანზე ყოველი ხდომილების წარმოქმნის შემდეგ და ხდება გამოთვლების შედარება ''ხელით'' ჩატარებულ რეზულტატებთან. ტრასირების დროს სასურველია პროგრამის ყოველი განშტოების შემოწმება. უმრავლესობა სიმულაციური მოდელებისა იძლევიან ტრასირების ჩატარების საშუალებას

პაკეტული ტრასირებისას ხშირად ვღებულობთ გამოსავალი მონაცენების დიდ მოცულობას, რომელიც უნდა შემოწმდეს შეცდომების აღმოჩენისათვის. ამიტომ უკეთესია ე.წ. *ინტერაქტიული გამართვის* ჩატარება.

ინტერაქტიული გამმართველი ანალიტიკოსს საშუალებას აძლევს შეაჩეროს მოდელის გატარება წინასწარ არჩეულ მომენტში, რათა შევისწავლოთ ან შევცვალოთ ცვლადების მნიშვნელობები.

მაგალითი 6.1. ნახ. 6.2-ზე ნაჩვენებია მმსმეკ-ის მუშაობის ტრასირება მისი ინტუიციური ახსნისთვის. პირველი სტრიქონი აღწერს სისტემას, ინიციალიზაციის შემდეგ $t = 0$ მომენტში, მეორე სტრიქონი აღწერს სისტემას პირველი ხდომილების დადგომის შემდეგ (მოთხოვნის დადგომა) და ა.შ.

ნახ. 6.2. მმსმეკ-ის ნაწილობრივი ტრასირება

ხდომილება	საათი	კვანძის მდგომარეობა	რიგში მოთხოვნათა რიცხვი	დადგომის დრო	ხდომილებათა სია	რიგში დაკავებულ მოთხოვნათა რიცხვი	საერთო დაკავება	ფუნქციის ქვეშა ფართობი, რიგში მოთხოვნათა რიცხვის განსაზღვრისათვის	ფუნქციის ქვეშა ფართობი, კვანძის დაკავებულობის განსაზღვრისათვის
ინიციალიზაცია	0	0	0		0,4	0	0	0	0
დადგომა	0,4	1	0		1,6 2,4	1	0	0	0
დადგომა	1,6	1	1	1,6	2,1 2,4	1	0	0	1,2
დადგომა	2,1	1	2	1,6; 2,1	3,8 2,4	1	0	0,5	1,7
გასვლა	2,4	1	1	2,1	3,8 3,1	2	0,8	1,1	2,0
გასვლა	3,1	1	0		3,8 3,3	3	1,8	1,8	2,7
გასვლა	3,3	0	0		3,8	3	1,8	1,8	2,9

- ზოგიერთი სიმულაციური მოდელის შესწავლისას სასარგებლოა მოდელირების გამოსავალი მონაცემების წარმოდგენა *ანიმაციით.*
- რათა ვატაროთ ნაკლები დაპროგრამება, უნდა ვისარგებლოთ სიმულაციური მოდელირების კომერციული პაკეტებით, მაგრამ მათი გამოყენება მოითხოვს სიფრთხილეს, რამეთუ შეიძლება წარმოიქმნას *არაცხადი შეცდომები.*

§6.4. ვალიდაციის აწევის მეთოდები

ამ პარაგრაფში საუბარი იქნება *ვალიდაციის აწევასა* და სიმულაციური მოდელის *დასაჯერობაზე.*

I. სისტემაზე მაღალ ხარისხიანი ინფორმაციის შეგროვება

სიმულაციური მოდელირებისას პროექტის ავტორებმა უნდა ისარგებლონ ყველანაირი ხელმისაწვდომი ინფორმაციით, რომელიც კი სისტემაზე შეიძლება მიიღონ სხვადასხვა მეთოდებით:

ა) *კონსულტაციები სპეციალისტებთან.* საჭიროა ძალიან ფრთხილად აირჩეს ის სპეციალისტები, რომლებისგანაც მიიღებთ ინფორმაციას საკვლევ სისტემაზე;

ბ) *დაკვირვება სისტემაზე.* თუ საკვლევი სისტემის მსგავსი სისტემა უკვე არსებობს, მაშინ მასზე დაკვირვებით შეიძლება შეგროვდეს მონაცენები, რომელიც შემდგომ მოდელირებაში გამოიყენება. რადგან ინფორმაციას ვიღებთ იმ ადამიანებისგან, რომლებიც არ არიან სიმულაციური მოდელების შემქმნელები, მნიშვნელოვანია 4 პრინციპის დაცვა:

- მოდელის შემქმნელები აუცილებელია დარწმუნდნენ, რომ მოთხოვნები მონაცემებისთვის ზუსტად გადაეცათ იმ პირებს, რომლებიც უნდა წარუდგინონ მათ;
- მოდელის შემქმნელებმა უნდა გაიგონ, რომ სისტემაზე დაკვირვების დროს მიღებული მონაცემები არის პროცესი და არა უბრალოდ მახასიათებლები, როგორც აბსტრაქტული სიდიდეები;
- მოდელირების პროცესში აუცილებელი მონაცემები შეიძლება მოიპოვოთ მსგავსი სისტემის მუშაობიდან შედარების მიზნით;
- პროექტის ავტორებს ხშირად უწევთ საკუთარი გამოცდილებისა და ინტუიციის გამოყენება, რათა ააგონ ჰიპოთეზები, თუ როგორ იმუშევებენ სიმულაციური მოდელის ცალკეული კომპონენტები.

II. რეგულარული ურთიერთქმედება მენეჯერთან

სიმულაციური მოდელირებით სისტემის კვლევის პროცესში ძალიან მნიშვნელოვანია, რომ მოდელის პროექტის ავტორები ურთიერთმოქმედებდნენ მენეჯერებთან. რათა მაქსიმალურად გაზარდონ ალბათობა იმისა, რომ მათ მიერ აგებული მოდელი სრულად იქნება გამოყენებული გადაწყვეტილების მიღების პროცესში. ასევე აღვნიშნოთ, რომ მენეჯერთან თანამშრომლობა მენეჯერს შეუქმნის ინტერესს პროექტის მიმართ და მისი პროექტში თანამონაწილეობა მეტ პასუხისმგებლობას წარმოშობს მოდელის რეალურ გამოყენებაზე. მოდელი გახდება უფრო დასაჯერები და სარწმუნო, თუ მენეჯერი მხარს უჭერს იმ დაშვებებსა და დეტალიზაციას, რომელიც ჩაიდო მოდელის აგებაში.

III. დეტალიზაციის დაშვებათა დოკუმენტური უზრუნველყოფა და სტრუქტურიზებული კრიტიკული ანალიზი

სასარგებლოა გაკეთდეს ყველა *ჩანაწერი დეტალიზაციის დაშვებებზე*, რომელიც საერთო შეთანხმებით მიღებულია სიმულაციური მოდელის აგებისთვის. ასეთ *ანგარიშს დაშვებათა დოკუმენტი* ეწოდება, მეორენაირად *კონცეპტუალური მოდელი.* ის ისე უნდა შედგეს, რომ მასთან მუშაობა შეეძლოთ ინჟინრებს, მენეჯერს და ა.შ. მასში წარმოდგენილი უნდა იყოს:

- მომიხილვითი ნაწილი, რომელშიც ფორმულირებული იქნება პროექტის ზოგადი ამოცანა; კონკრეტული პრობლემები, რომლებიც განიხილება სისტემის შესწავლისას მოდელირებით; მუშა მახასიათებლები გამოთვლებისთვის;
- ყოველი ქვესისტემის სრული აღწერა *წინასწარ ფორმატში.* ამ ქვესისტემებს შორის ურთიერთქმედება;
- რა გამარტივებები იქნა გაკეთებული და რატომ?
- მონაცემთა მოკლე აღწერა (ისეთების, როგორიცაა შერჩევითი საშუალო, მონაცემთა ჰისტოგრამა, ალბათური განაწილება, რომელიც მორგებული იქნება მოცემულ მონაცემებზე);
- მნიშვნელოვანი და წინააღმდეგობრივი ინფორმაციის წყაროები;
- ანგარიშის შექმნისას აუცილებელია გვახსოვდეს, რომ ანგარიში შეთანხმებული უნდა იყოს მენეჯერთან და ის ანგარიშში კარგად უნდა ერკვეოდეს.

რადგან ინფორმაციის წყარო სხვადასხვა პროფესიის ადამიანებია და მათ არა აქვთ დიდი დრო სრულად წარმოადგინონ მათი ცოდნა ან მონაცემები საკვლევ სისტემაზე, ცხადია მათგან მიღებული ინფორმაცია არასრულია, არაზუსტია და ხშირად წინააღმდეგობრივიც. ამ პრობლების ერთ-ერთი გადაწყვეტაა *კონცეპტუალური მოდელის სტრუქტურული გარჩევის ჩატარება.* შედგება შეხვედრა და მიღებული იქნება დოკუმენტი, რომელშიც წარმოდგენილი იქნება დეტალიზაციაზე და არა მარტო მასზე *დაშვებათა განხილვა* შესასწავლი ამოცანების სპეციალისტებისა და მენეჯერების თანხლებით. ინფორმაციის გაცვლას, რომელიც ხდება ამ შეხვედრაზე, აქვს ძალიან დიდი მნიშვნელობა. დაშვებებზე დოკუმენტის (ანგარიშის) სტრუქტურიზებულ კრიტიკულ ანალიზს *კონცეპტუალური მოდელის ვალიდაცია* ეწოდება. გვახსოვდეს, რომ ეს ყველაფერი *ტარდება დაპროგრამების დაწყებამდე!*

ძალიან მნიშვნელოვანია სიმულაციური მოდელის *გამოსავალი მონაცენების ვალიდაცია. სიმულაციური მოდელის ადექვატურობის შემოწმება დასრულდება* იმ დასტურით, რომ მისი გამოსავალი მონაცენები *იდენტურია* იმ გამოსავალი მონაცენებისა, რომელსაც უნდა ველოდოთ რეალური, საკვლევი სისტემიდან. ვალიდაციის ასეთი რაოდენობრივი მეთოდები დაწვრილებით განხილულია [1]-ში.

თავი 7. სხვადასხვა ამოცანები სიმულაციასა და მოდელირებაზე (ჯგუფური პროექტის ამოცანები)

წინა თავების გაცნობით ჩვენ შევიძინეთ გარკვეული ცოდნა რთული სისტემების სიმულაციური მოდელირების ეტაპების შესრულებაზე (სიმულაციური მოდელის პროექტის შესრულების ეტაპები):

- საკვლევი სისტემის სიმულაციური მოდელის პროექტის კონსტრუირება;
- სიმულაციური მოდელის დაპროგრამება;
- პროგრამის ტესტირება;
- პროგრამის გამართვა სამუშაო მდგომარეობაზე;
- შესრულებული პროექტის პრეზენტაცია და ანგარიშების წარმოდგენა.

გთავაზობთ რამოდენიმე საინტერესო ამოცანას სიმულაციურ მოდელირებაზე. თითოეული ამოცანის რეალიზაცია შეასრულეთ *ჯგუფური პროექტის* სახით (რამოდენიმე, 3 ან 4 სტუდენტის ერთობლივი მონაწილეობით), სადაც გათვალისწინებული იქნება აქ წარმოდგენილი ოთხივე ეტაპი და სიმულაციური მოდელირების ყველა შემდეგი ნიუანსი:

- ყოველი ამოცანა ფაქტიურად წარმოადგენს საკვლევი სისტემის *დეტალიზაციას,* თუმცა მისი დაზუსტება შესაძლებელია *პროგრამის ხელმძღვანელთან* (სისტემის სიმულაციური მოდელის *დამკვეთთან,* რომელიც ასევე სისტემის *ექსპერტიც* იქნება);
- ყოველ ჯგუფურ პროექტს ეყოლება *კონსულტანტი-პროგრამისტი*;
- ყოველ ჯგუფს ეყოლება *ლიდერი* (სიმულაციის პროექტის კონსტრუქტორ-მენეჯერი, არჩეული ჯგუფის წევრების მიერ). იგი ხელმძღვანელობს სიმულაციის პროექტის აგებას და ჯგუფის მონაწილეებზე ფუნქციათა განაწილებას, თუმცა კი ჯგუფური პროექტის წარმატებაზე თანაბარი პასუხისმგებლობა აკისრია ჯგუფის ყოველ წევრს. ლიდერი წარმართავს დიალოგს დამკვეთან და კონსულტატთან, გეგმავს და მათთან შეხვედრებს ატარებს ინდივიდუალურად და ჯგუფთან ერთად; ხელმძღვნელობს საპრეზენტაციო მოხსენების მომზადებას შესრულებულ პროექტზე;
- დასრულებული პროექტი პრეზენტაციაზე წარმოდგენილი უნდა იყოს პროექტის ფარგლებში შეთანხმებული *ანგარიშების ფორმებით*;
- პრეზენტაციაზე წარმოდგენილი ნამუშევრის ვალიდაციისა და ვერიფიკაციის ხარისხს აფასებს *დამკვეთი, მოწვეული ექსპერტი და კონსულტანტი.*

§7.1. დამთხვევის ამოცანა და მისი სიმულაცია

შემთხვევითად არეული თითოეული N რაოდენობის მქონე ბანქოს ორი კოლოდა ერთმანეთს დარდება შემდეგი წესით. პირველი ბანქო – პირველს, მეორე ბანქო – მეორეს, და ა.შ. N-ური ბანქო – N-ურს. თუ ბანქოები რაიმე k–ურ ადგილზე თანხვდებიან, მაშინ იტყვიან, რომ ადგილი აქვს *დამთხვევას.* ვთქვათ A_k აღნიშნავს k-ურ ადგილზე დამთხვევას. განვიხილოთ ხდომილება:

$$A = A_1 + A_2 + \cdots + A_N$$

ერთი მაინც დამთხვევა ბანქოს 2 კოლოდაში. ცნობილია, რომ ამ ხდომილების ალბათობა გამოითვლება ფორმულით [7]:

$$p = \Pi(A) = 1 - \frac{1}{2!} + \frac{1}{3!} - + \cdots \pm \frac{1}{N!}. \qquad (7.1)$$

გავიხსენოთ e^x -ექსპონენციალური ფუნქციის ხარისხოვან მწკრივად გაშლა, როცა $x = -1$:

$$e^{-1} = 1 - 1 + \frac{1}{2!} - \frac{1}{3!} + + \cdots \pm \frac{1}{N!} + \cdots.$$

მაშასადამე გვაქვს (7.1)-ის მიახლოებითი შეფასება:

$$p \approx 1 - e^{-1}. \qquad (7.2)$$

ფორმულა (7.2) იმდენად ზუსტია, რამდენადაც N დიდია. ამას ადასტურებს შემდეგი ცხრილიც:

N	3	4	5	6	7
p	0.66667	0.62500	0.63333	0.63196	0.63214

ეს ამოცანა ტოლფასია პრაქტიკაში ხშირად წარმოქმნილი *სახუმარო მოვლენებისა. მაგალითად*: წარმოდგენის პერიოდში თეატრის საგარდერობოში თანამშრომელთა ოინბაზმა ბავშვებმა მაყურებელთა ქუდები "შემთხვევითი წესით" შეცვალეს. რა არის ალბათობა იმისა, რომ ერთი მაინც მაყურებელი მიიღებს თავის კუთვნილ ქუდს წარმოდგენის შემდეგ?

ეს ალბათობა დაახლოებით (7.2) ფორმულით გამოითვლება და ყველაზე საინტერესო ის არის, რომ ალბათობა თითქმის არაა დამოკიდებული N -ზე მის ზრდასთან ერთად და დაახლოებით $\frac{2}{3}$-ის ტოლია $\left(1 - e^{-1} \approx \frac{2}{3}\right)$!

სიმულაცია დამთხვევის ამოცანაში

შევადგინოთ C-პროგრამა და მოვახდინოთ $\{1, 2, \ldots, N\}$ სიმრავლის $N!$ რაოდენობა თანაბარი განაწილების სიმულაცია:

1 ბიჯი: γ_1 შემთხვევითი რიცხვით გათამაშდება n_1 ნომერი – ადგილი $\{1, 2, \ldots, N\}$ სიმრავლეში;

2 ბიჯი: γ_2 შემთხვევითი რიცხვით გათამაშდება n_2 ნომერი – ადგილი $\{1, 2, \ldots, N\}$-ში, $n_1 \neq n_2$;

და ა.შ.,

$N-1$ ბიჯი: γ_{N-1} შემთხვევითი რიცხვით გათამაშდება n_{N-1} ნომერი – ადგილი $\{1, 2, \ldots, N\}$-დან დარჩენილ 2 ადგილზე;

N ბიჯი: n_N ნომერს გათამაშება არ უნდა. ის დაიკავებს ბოლო, დარჩენილ ადგილს;

მიიღება $\{n_1, n_2, \ldots, n_N\}$ ინვერსია.

შეადარეთ $\{1, 2, \ldots, N\}$ და $\{n_1, n_2, \ldots, n_N\}$ ინვერსიები. თუ ერთი კომპონენტური დამთხვევა მაინც იქნება, მთვლელი S გაიზრდება 1–ით (თავიდან ის 0–ის ტოლია). თუ სიმულაცია ინვერსიების შედარებაზე ტარდება L -ჯერ, მაშინ ერთი მაინც დამთხვევის p ალბათობა შეფასდება $\hat{p} = \frac{S}{L}$ სიხშირით.

შევადაროთ $\frac{S}{L}$ და $1 - e^{-1}$ რიცხვები:

სიმულაციის გამოყენებად შეიძლება ჩაითვალოს ის, რომ N -ისა და L -ის გაზრდით შეგვიძლია მაღალი სიზუსტითი გამოვთვალოთ *ნეპერის რიცხვი*:

$$e \approx \frac{L}{L-S}.$$

კონკრეტული ამოცანები ნეპერის რიცხვის მაღალი სიზუსტით გამოთვლაზე

სიმულაციით შეავსეთ ცხრილი და შეადარეთ e-ს მისი შეფასებები

ამოცანის ნომერი	N	L	S	$e=\frac{L}{L-S}$	შეფასებაში რამდენი თანრიგია სწორი მძიმის შემდეგ?
1	6	30			
2	6	50			
3	6	70			
4	10	30			
5	10	70			
6	10	70			
7	10	100			

§7.2. შემთხვევითი ხეტიალი – გაკოტრების ამოცანა და მისი სიმულაცია

ამოცანა 7.2.1. ვთქვათ თამაშში მონაწილეობს ორი მოთამაშე, რომელთაგან ერთი (ჩვენი) მოთამაშე იგებს და აგებს 1$-ს შესაბამისად p და q ალბათობებით $(p+q=1)$. ვთქვათ მისი საწყისი კაპიტალია z\$. ხოლო მისი მოწინააღმდეგის საწყისი კაპიტალი კი $(a-z)$\$. მათი ჯამური საწყისი კაპიტალია a\$. თამაში გრძელდება მანამ, სანამ ჩვენი მოთამაშის კაპიტალი გახდება ან 0 (გაკოტრდება) ან a (მოიგებს). ე.ი. თამაში გრძელდება *ერთ-ერთი მოთამაშის გაკოტრებამდე.*

გაკოტრების კლასიკური ამოცანა:

როგორ გამოითვლება ჩვენი მოთამაშის *გაკოტრების ალბათობა* q_z?

პასუხი [7]:

$$q_z = \begin{cases} \dfrac{\left(\frac{q}{p}\right)^a - \left(\frac{q}{p}\right)^z}{\left(\frac{q}{p}\right)^a - 1}, & \text{თუ} \quad q \neq p, \\ 1 - \dfrac{z}{a}, & \text{თუ} \quad q = p. \end{cases} \tag{7.3}$$

ეს ამოცანა შეიძლება ფორმულირებული იყოს მისი ტოლფასი შემდეგი ამოცანის სახით.

ამოცანა 7.2.2. ვთქვათ მოთამაშე z საწყისი კაპიტალით ეთამაშება "უსასრულოდ" მდიდარ მეტოქეს, რომელიც ყოველთვის მზადაა გააგრძელოს თამაში, მაშინ როდესაც ჩვენ მოთამაშეს აქვს უფლება შეწყვიტოს თამაში მისი ნებით შემდეგი სტრატეგიით: იგი თამაშობს მანამ, სანამ მისი კაპიტალი გახდება ან 0 (გაკოტრდება) ან a ($a-z$ სუფთა მოგებით). მაშინ (7.3)-ით მოცემულია მისი გაკოტრების ალბათობა, ხოლო $1-q_z$ კი მისი მოგების ალბათობაა. ჩვენი მოთამაშის მახასიათებელი *მოგება-წაგების* შემთხვევითი სიდიდეა:

$$G \sim \begin{pmatrix} -z & a-z \\ q_z & 1-q_z \end{pmatrix},$$

საიდანაც ჩვენი მოთამაშის *მოსალოდნელი მოგებაა* $\mathrm{E}(G)=(a-z)(1-q_z)-q_z z$. გამარტივებით

$$\mathrm{E}(G)=a(1-q_z)-z. \tag{7.4}$$

ცხადია, რომ $\mathrm{E}(G)=0$ მაშინ და მხოლოდ მაშინ, როცა $q=p$, ანუ როცა თამაში ორივე მხარისთვის *სამართლიანია.*

ამოცანა 7.2.3. ცხადია, რომ თამაშის ხანგრძლივობა 7.2.1 და 7.2.2 ამოცანებში შემთხვევითი სიდიდეა. აღვნიშნოთ მისი მათემატიკური ლოდინი D_z-ით.

D_z – *თამაშის ხანგრძლივობის* მოსალოდნელი მნიშვნელობაა (იზომება თამაშების რაოდენობით) [7]:

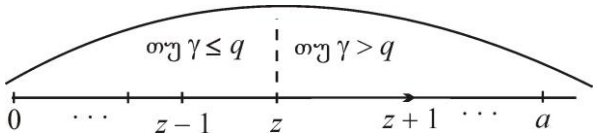
$$D_z=\begin{cases}\dfrac{z}{q-p}-\dfrac{q}{q-p}\cdot\dfrac{1-\left(\frac{q}{p}\right)^z}{1-\left(\frac{p}{q}\right)^a}, & \text{თუ} \quad p\neq q,\\[2ex] z(a-z), & \text{თუ} \quad p=q.\end{cases} \tag{7.5}$$

ამოცანა 7.2.4. (გაკოტრება n-ურ ბიჯზე): ვთქვათ $U_{z,n}$ – ალბათობაა იმისა, რომ ჩვენი მოთამაშისთვის თამაში გაკოტრებით დასრულდება თამაშის n-ურ ბიჯზე. ანუ ჩვენი მოთამაშე გაკოტრდება n-ურ თამაშზე: მიუხედავად უამრავი გზებისა, თუ როგორ ამოვხსნათ ეს ამოცანა, მარტივი გამოსათვლელი ფორმულა არ არსებობს. მაგალითადმ [7]:

$$U_{z.n}=a^{-1}2^n p^{\frac{n-z}{2}} q^{\frac{n+z}{2}}\sum_{\nu=1}^{a-1}\left[\cos^{n-1}(\pi\nu/a)\sin(\pi\nu/a)\sin(\pi\nu/a)\right]. \tag{7.6}$$

გაკოტრების კლასიკური მოდელის სიმულაცია:

გაათამაშეთ გაკოტრების მოდელი 7.2.1.-7.2.4 ამოცანებისთვის. შეადგინეთ C-პროგრამა. მთავარი სიმულაცია ასეთია (ხშირად ამ გეომეტრიული ინტერპრეტაციის გამო გაკოტრების ამოცანა ფორმირდება *შემთხვევითი ხეტიალის* ამოცანებში):



ნახ. 1

7.2.1, 7.2.2, 7.2.3 და 7.2.4 ამოცანების ალბათობებისა თუ მოსალოდნელი მნიშვნელობების შეფასება სიმულაციის გზით მარტივია სათანადო *მთვლელების* საშუალებით. პროგრამის პარამეტრებია: p, q, z, a ; n – თამაშების რაოდენობა; N – სიმულაციათა სერიების რიცხვია. შეფასებები შეადარეთ ზუსტ 7.3-7.6 ფორმულებს.

ამოცანა 7.2.5. (განზოგადოებული შემთხვევითი ხეტიალის სქემა) განსხვავებით წინა ამოცანებისა, დავუშვათ, რომ ერთ თამაშში შესაძლებელია თანხის არა ერთით შემცირება ან ერთით გაზრდა, არამედ k მთელი რიცხვით. განვიხილოთ გაკოტრების ამოცანის შემდეგი ვარიანტი: მოთამაშეს აქვს z კაპიტალი. ის საბოლოოდ კოტრდება ან მოიგებს ($a-z$ სუფთა მოგებით). ითვლება, რომ ერთეულოვან თამაშში შესაძლებელია $k\in Z$ კაპიტალის შეძენა. ამ ამოცანის ფორმალიზება ასე შეიძლება: n თამაშის შემდეგ ჩვენი მოთამაშის კაპიტალი იქნება:

$$x = z + X_1 + X_2 + \cdots + X_n\,, \tag{7.7}$$

სადაც $X_1, X_2, \ldots, X_n$ – ერთნაირად განაწილებულ მთელმნიშვნელობიანი დამოუკიდებელი შემთხვევითი სიდიდეებია $(P(X_i = \nu) \equiv P_\nu)$. თამაში სრულდება მაშინ, როცა

$x = z + X_1 + X_2 + \cdots + X_n \le 0$ (მოთამაშე გაკოტრდა) ან

$z + X_1 + X_2 + \cdots + X_n \ge a$ (მოთამაშე მოგებულია).

ამოცანა მდგომარეობს U_z – *გაკოტრების ალბათობის* შეფასებაში.

პირველ თამაშზე მოთამაშის გაკოტრების ალბათობაა:

$$r_z = P_{-z} + P_{-z-1} + P_{-z-2} + \cdots$$

(ეს სიდიდე შეიძლება იყოს 0-ის ტოლიც). ერთეულოვან თამაშში მოთამაშის z -დან $x \in (0, a)$ კაპიტალზე გადასვლის ალბათობაა P_{x-z}, ხოლო x -დან გაკოტრების ალბათობაა U_x. მაშინ U_z გაკოტრების ალბათობა (სრული ალბათობის ფორმულით) აკმაყოფილებს შემდეგ განტოლებას:

$$U_z = \sum_{x=1}^{a-1} U_x \cdot P_{x-z} + r_z\,. \tag{7.8}$$

ამოცანის ამოხსნა ნიშნავს $\{U_z\}$ ცვლადების მიმართ $(z = 1, 2, \ldots, a-1)$ (7.8) წრფივი განტოლებათა სისტემის ამოხსნას, რაც საკმაოდ რთულდება a -ს ზრდასთან და $\{P_\nu\}_\nu$ განაწილებაში დადებითი მნიშვნელობების მატებასთან ერთად. ცხადია *ერთადერთი გამოსავალი რჩება – სიმულაცია.* რაც ნიშნავს X_i შემთხვევითი სიდიდეების მიმდევრობის გენერირებას და სათანადო მთვლელებით $\{U_z\}_z$ -განაწილების შეფასებას.

7.2.4 ამოცანის მსგავსად შეადგინეთ C-კოდი და შეაფასეთ გაკოტრების ალბათობა და მოსალოდნელი მნიშვნელობები.

ამ ამოცანის კონტექსტში განვიხილოთ მაგალითი, რომლის ალბათურ მოდელს წარმოადგენს გაკოტრების 7.2.5 ამოცანა:

ამოცანა 7.2.6. *ამოცანის დასმა*: საქონლის (ნაწარმის) პარტიის ხარისხის შემოწმებისთვის გამოიყოფა შემთხვევითი N მოცულობის შერჩევები, რომელზეც კეთდება ექსპერტიზა. დავუშვათ, რომ შერჩევები დამოუკიდებელია და თითოეულ შერჩევაში დაწუნებული საქონლის რაოდენობას ერთი და იგივე ბინომიალური განაწილება გააჩნია: ითვლება, რომ თავიდან ერთი დაწუნებული საქონელი შერჩევაში ყოველთვის იქნება $(z = 1)$. k -ურ შერჩევაში, თუ X_k დაწუნებული საქონლის რიცხვია, მაშინ

$$P_j = \Pi(X_k = j) = C_N^j \cdot p^j \cdot q^{N-j} \quad (j \in Z), \tag{7.9}$$

სადაც p დაწუნებული ნაწარმის (საქონლის) ალბათობაა $(q = 1 - p)$, ითვლება, რომ $P_j = 0$, თუ $j \le -1$ $(j \in Z)$.

ხარისხის შემოწმების მოდელის პროცედურული წესი ასეთია:

1. გაკეთდება I შერჩევა. თუ შერჩევა არ შეიცავს არც ერთ დაწუნებულ საქონელს, ჩაითვლება, რომ პარტია მისაღებია $(X_1 = 0)$;
2. თუ შერჩევაში წუნის რაოდენობა აღემატება a -ს, მაშინ მთელი პარტია დაწუნებულია $(X_1 > a)$;
3. თუ $x = 1 + X_1$ დაწუნებული საქონლის რაოდენობა $1 \le x \le a$ ინტერვალშია, მაშინ:

4. შერჩევების რიცხვი იზრდება მანამდე, სანამ დაწუნებული საქონლის რაოდენობა $x = 1 + X_1 + X_2 + \cdots + X_k > a$ -ანუ პარტია დაწუნებული იქნა, ან პარტია ვარგისია, თუ რომელიღაც $X_k = 0$;
5. შერჩევების სერია დასრულებულია (N -ჯერადი სერიის გავლის შემდეგ) გადაწყვეტილებით: ა) პარტია დაწუნებულია, ბ) პარტია ვარგისია.

პროცედურული წესი საშუალებას გვაძლევს შევაფასოთ პარტიის ხარისხი, მაგრამ ამ პროცედურამ შეიძლება დიდი *რაოდენობა დროისა და ფინანსების რესურსები მოითხოვოს.* ასეთ შემთხვევაში *ხარისხის წინასწარი შეფასების მეთოდები გამოიყენებაა.* მათ შორის გამორჩეულია *სიმულაცია*, რომელიც ხარისხის შემოწმების "ხელოვნურ" ექსპერიმენტს წარმოადგენს.

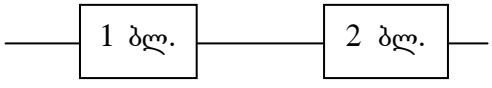
ერთერთი პარამეტრი, რომელსაც $X_k \sim \begin{pmatrix} 0,\ 1,\ 2,\ \ldots,\ N \\ p_0,\ p_1,\ p_2,\ \ldots,\ p_N \end{pmatrix}$ განაწილების სიმულაცია ითხოვს, p პარამეტრია, რომელიც როგორც წესი საქონლის მწარმოებელი ორგანიზაციის სტანდარტია (იგი წინასწარ შეფასებულია).

რიცხვითი მაგალითები: პროცედურული წესის საფუძველზე მიიღეთ გადაწყვეტილება საქონლის ხარისხზე სიმულირებით, თუ ხარისხის შემოწმების პარამეტრები ჩაწერილია ცხრილის სახით (შეადგინეთ C-კოდი და ყოველი მაგალითისთვის შეავსე:

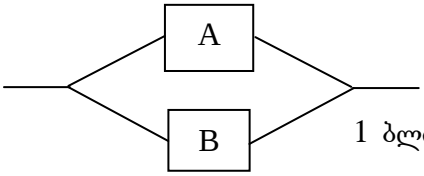
i – მაგალითის №	N – შერჩევის მოცულობა	a – პარამეტრი (მოწმარებლის პარამეტრი), 1-a/N – წუნის წილი (დასაშვებია პარტიაში)	p – წუნის პროცენტი (მწარმოებლის პარამეტრი)	p -ის და $1-\frac{a}{N}$ -ს შედარება	გადაწყვეტილების მიღება: წ – წუნია (დეფექტი), ვ – ვარგისია (კარგია)	სიმულაციის რამდენი სერია დაჭირდა გადაწყვეტილების მიღებას?
7.2.6.1	100	80	0.2			
7.2.6.2	500	460	0.2			
7.2.6.3	300	270	0.1			
7.2.6.4	50	450	0.1			
7.2.6.5	1000	920	0.08			
7.2.6.6	2000	1950	0.02			

§7.3. ზოგიერთი სისტემის მუშაობის საიმედობის შეფასება სიმულაციით

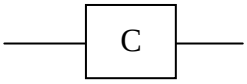
7.3.1. ვთქვათ სისტემა შედგება მიმდევრობით შეერთებული



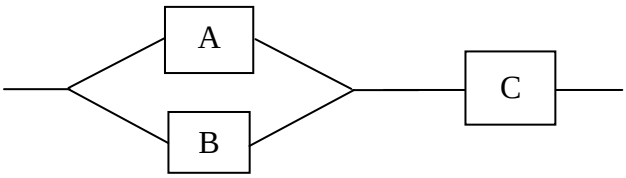
ორი ბლოკისგან. სისტემა გამოდის მწყობრიდან, თუ მწყობრიდან გამოდის მისი ერთი ბლოკი მაინც. პირველი ბლოკი შედგება ორი A და B ელემეტებისგან, რომლებიც შეერთებულია პარალელურად. იგი გამოდის მწყობრიდან,



როდესაც მწყობრიდან გამოდის ორივე ელემენტი. მეორე ბლოკი შედგება ერთი C ელემენტისგან.



იგი გამოდის მწყობრიდან, როცა მწყობრიდან გამოდის C ელემენტი. სისტემის მოდელის წარმოდგენა ელემენტებში ასეთია.



A – ელემენტის საიმედობის, უწყვეტად მუშაობის ალბათობაა $P(A)=0.8$.
B – ელემენტის საიმედობის, უწყვეტად მუშაობის ალბათობაა $P(B)=0.85$.
C – ელემენტის საიმედობის, უწყვეტად მუშაობის ალბათობაა $P(C)=0.6$.

ა) შეადგინე C-პროგრამა და სიმულაციით შეაფასე სისტემის P_N^* – საიმედობა ($P_N^*=\frac{k}{N}$, სადაც N სიმულიაციათა რიცხვია, k – N სიმულაციაში საიმედოდ მუშაობის რიცხვი), სისტემის უწყვეტად მუშაობის ალბათობის შეფასება.

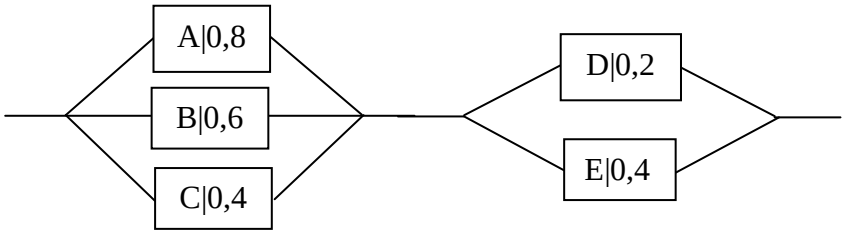
ბ) იპოვე სისტემის საიმედობის ზუსტი P მნიშვნელობა და შეაფასე აბსოლუტური ცდომილება $\left|P_N^*-P\right|$, $N=20, 40, 60$.

სიმულაციის №	ბლოკის №	სისტემის ელემენტების მუშაობის მოდელირების შემთხვევითი რიცხვები			დასკვნები სისტემის საიმედოდ მუშაობაზე				
					ელემენტები			ბლოკები	სისტემა
		A	B	C	A	B	C		
1	1 2	0,10	0,09	 0,73	+	+	 –	+ –	–
2	1 2	0.25	0.33	 0,76	+	+	 –	+ –	–
3	1 2	0,52	0,01	 0,35	+	+	 +	+ +	+
4	1 2	0,86	0,34	 0,67	–	+	 –	+ –	–

შენიშვნა: სისტემის სიმულაციის შედეგები ცხრილის სახით შეიძლება ასე წარმოვადგინოთ, რაც დაგეხმარებათ პროგრამის შედგენასა და საბოლოო ამოცანის გადაწყვეტაში. ცხრილში 0.10, 0.09, 0.73, . . . , 0.86, 0.34, 0.67 – შემთხვევითი რიცხვების რაიმე მიმდევრობაა.

შეადგინე C-კოდი, ასეთი გათამაშებით განახორციელეთ სისტემის სიმულაცია $N = 20$, 40, 60-თვის.

7.3.2. ვთქვათ სისტემა შედგება შემდეგი 2 ბლოკისგან და 5 ელემენტისგან.



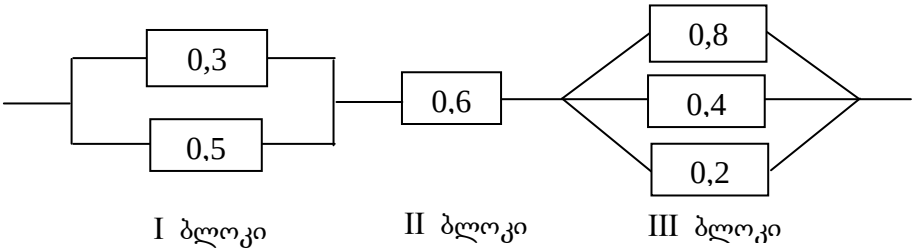
ა) შეადგინე C-პროგრამა და სიმულაციით შეაფასე P_N^* სისტემის საიმედობა $(N = 20)$;

ბ) გამოთვალე სისტემის ზუსტი P –საიმედობა და სიზუსტე $\left|P - P_N^*\right|$;

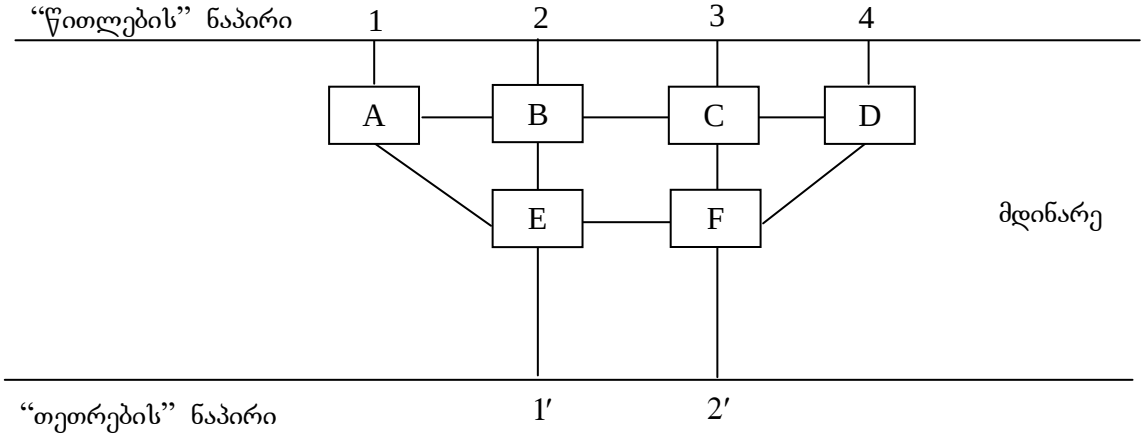
გ) გაზარდე $N = 40$ -მდე სიმულაციათა რიცხვი და კვლავ შეაფასე სიზუსტე. რა მოუვიდა სიზუსტეს?

დ)* ა) ამოცანისთვის II ბლოკში სულ მცირე რამდენი ელემენტი უნდა ჩავუმატოთ 0.2 ალბათობით, რომ საიმედობა 20%–ით გაიზარდოს? $(N = 40)$.

7.3.3. იგივე ამოცანა, რაც 7.3.2.



7.3.4. (მდინარის ფორსირების ამოცანა):



ნახაზზე მოცემულია მდინარეზე კონსტრუირებული სამხედრო დანიშნულების ტივების (“ბარჟების”) (A, B, C, D, E, F) კონსტრუქცია-შეერთება, რომელიც საომარი ტექნიკის თუ სამხედრო ძალების საკომუნიკაციო-გადასასვლელ გზას წარმოადგენს. “საომარი მოქმედების” სიტუაცია ასეთია. “წითლებს” სურთ მდინარის ფორსირება და გადასვლა “თეთრების” ნაპირზე. “თეთრები” ცდილობენ არტილერიით საკომუნიკაციო კონსტრუქციის დაზიანებას (მოქმედების დრო მინიმალურია).

“თეთრების” მიერ თითოეული ბარჟის დაზიანების ალბათობაა ½ . ბარჟებს შორის დახრილი მონაკვეთის დაზიანების ალბათობაა 1/5, ხოლო ვერტიკალური მონაკვეთების დაზიანების ალბათობა კი – 1/10 (საომარი მოქმედების დრო მინიმალურია).

კონსტრუქცია ითვლება დაზიანებულად, თუ არ იარსებებს გადასასვლელად გზა ერთი ნაპირიდან მეორე ნაპირზე (სისტემას არ შეუძლია ნაკადის გატარება).

ა) შეადგინე C-პროგრამა და სიმულაციით შეაფასე P_N^* – კონსტრუქციის საიმედობა ($N=20$, $N=30$ შემთხვევაში), რომელიც განისაზღვრება, როგორც 1-ს მინუს კონსტრუქციის დაზიანების ალბათობა.

ბ) დავრწმუნდეთ P – ზუსტი საიმედობის დათვლის შრომატევადობაში და მისი გამოთვლის არაპრაქტიკულობაში.

7.3.5.* წინა ამოცანის საკომუნიკაციო კონსტრუქციისთვის საომარი მოქმედების პერიოდში “ბარჟების” უწყვეტად ფუნქციონირების განაწილება ექსპონენციალურია პარამეტრებით:

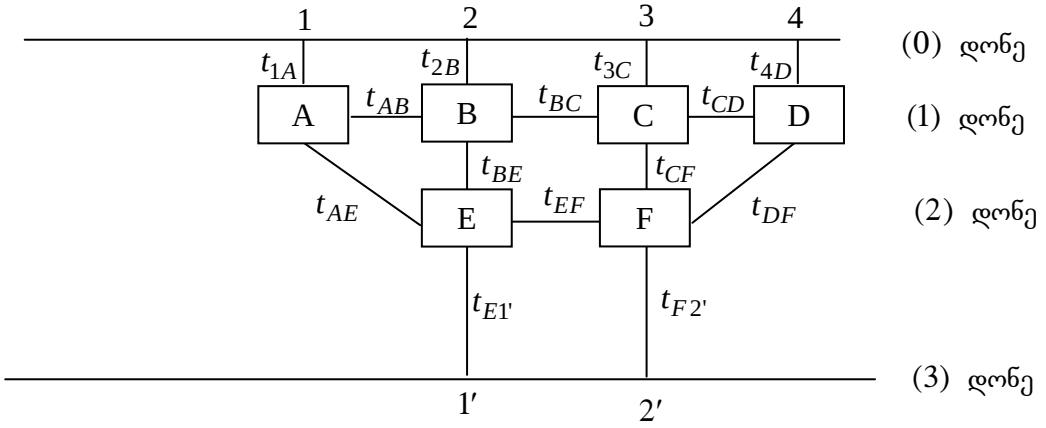
$$\lambda_A=\lambda_B=\lambda_C=\lambda_D=0{,}03\,(1/\text{წთ}),\ \text{ხოლო}\ \ \lambda_E=\lambda_F=0{,}02\,(1/\text{წთ}).$$

დახრილი მონაკვეთების მუშაობის განაწილებაც ექსპონენციალურია $\lambda_{\backslash}=\lambda_{/}=0.05\,(1/\text{წთ})$ და ვერტიკალური მონაკვეთების მუშაობის ინტენსივობაა $\lambda_{|}=0.08\ \ (1\,\text{წთ})$.

ა) შეადგინეთ C-პროგრამა და სიმულაციებით შეაფასეთ გადასასვლელი კონსტრუქციის საიმედობა $P^*=P^*(N)$ (1/4 სთ-ის განმავლობაში; 1/2 სთ-ის განმავლობაში, 1 სთ-ის განმავლობაში);

ბ) გამოიკვლიეთ და შეაფასეთ სისტემის უწყვეტად მუშაობის (სისტემას შეუძლია ნაკადის გატარება) საშუალო დრო (ჩაატარეთ $N=50,100,150,\ldots$ სიმულაცია) $T^*=T^*(N)$. N –ზე დამოკიდებულების მონაცემები წარმოადგინეთ გრაფიკულად.

შენიშვნა: მოდელის შექმნისა და სიმულაციის ჩატარებისთვის შევქმნათ *კონსტრუქცია-პროექტი* (უწყვეტად მუშაობის სქემა ბარჟებისა და მათი შემაერთებელი მონაკვეთების მიხედვით):



სადაც $t_{\bullet}$ აღნიშნავს მოცემული ობიექტის უწყვეტად მუშაობის დროს. t აღნიშნავს სისტემის უწყვეტად მუშაობის დროს.

ვთქვათ “+” – მაქსიმუმია ოპერაციია, “·“ – მინიმუმის ოპერაციაა.

სისტემის საიმედობა ეწოდება t-ს მათემატიკური ლოდინს – $\mathrm{E}(t)$-ს.

1. $t_{1A}, t_{2B}, t_{3C}, t_{4D}, t_{BE}, t_{CF}, t_{DF}, t_{E1'}, t_{F2'}$ შემთხვევითი სიდიდეები განაწილებულია ექსპონენციალურად $\lambda=0.08\left(\frac{1}{\text{წთ}}\right)$ ინტენსივობით.

2. t_{AE} და t_{DF} განაწილებულია ექსპონენციალურად $\lambda=0.05\left(\frac{1}{\text{წთ}}\right)$ ინტენსივობით.

3. t_A, t_B, t_C, t_D – განაწილებულია ექსპონენციალურად $\lambda = 0.03\left(\frac{1}{წთ}\right)$, ხოლო t_E და t_F კი $\lambda = 0.02\left(\frac{1}{წთ}\right)$ ინტენსივობით.
4. ცხადია ბარჟების დაზიანების ინტენსივობა უფრო მაღალია, ვიდრე მათი შემაერთებელი მონაკვეთების, რადგან ისინი უფრო ინტენსიურად "დაიბომბება" მოწინააღმდეგის მიერ.
5. t-ს განაწილების გამოთვლა და მაშასადამე სისტემის საიმედობისაც პრაქტიკულად შეუძლებელია, რადგან t-ს გამოსახულება საკმაოდ რთულია:
$$t = t^{(01)} \cdot \left(t_A^{(12)} + t_B^{(12)} + t_C^{(12)} + t_D^{(12)}\right) \cdot \left(t_E^{(23)} + t_F^{(23)}\right), \qquad (*)$$
სადაც $t^{(01)}$ – 0-დან 1 დონეზე გადასვლის დროა,
$$t^{(01)} = t_{1A} + t_{2B} + t_{3C} + t_{4D};$$
$t_A^{(12)}$ – A ბარჟიდან მე-2 დონეზე გადასვლის დროა,
$$t_A^{(12)} = t_A(t_{AE} + t_{AB} \cdot t_B \cdot t_{BE}) \cdot t_E + t_A \cdot t_{AB} \cdot t_{BC} \cdot t_C(t_{CF} + t_{CD} \cdot t_D \cdot t_{DF})t_F;$$
ანალოგიურად
$$t_D^{(12)} = t_D(t_{DF} + t_{CD} \cdot t_C \cdot t_{CF}) \cdot t_F + t_D \cdot t_{CD} \cdot t_C \cdot t_{BC} \cdot t_B(t_{BE} + t_{AB} \cdot t_A \cdot t_{AE})t_E;$$
$$t_C^{(12)} = t_C(t_{CF} + t_{CD} \cdot t_D \cdot t_{DF}) \cdot t_F + t_C \cdot t_{BC} \cdot t_B(t_{BE} + t_{AB} \cdot t_A \cdot t_{AE})t_E;$$
$$t_B^{(12)} = t_B(t_{BE} + t_{AB} \cdot t_A \cdot t_{AF}) \cdot t_E + t_B \cdot t_{BC} \cdot t_C(t_{CF} + t_{CD} \cdot t_D \cdot t_{DF})t_F.$$
6. t-ს *გამოსახულების მოდელირება სიმულაციით* გულისხმობს მასში შემავალი ყოველი კომპონენტის – შემთხვევითი სიდიდის გათამაშებას ექსპონენციალური კანონიდან ($1 - e^{-\lambda t_{\exp}} = \gamma \Rightarrow e^{-\lambda t_{\exp}} = 1 - \gamma$, $t_{\exp} = -\frac{1}{\lambda}\ln(1-\gamma)$, $1-\gamma$ და γ თანაბარი განაწილებისაა $(0,1)$-ზე. ამიტომ $1-\gamma \sim \gamma$ და $t_{\exp} = -\frac{1}{\lambda}\ln(1-\gamma)$), მათი t-ს გამოსახულებაში ჩასმას და t-ს გამოთვლას.
7. ა) ამოცანაში N ჩატარებული სიმულაციიდან დაითვლება რამდენ შემთხვევაში $t \geq \frac{1}{4}$ სთ-ზე და ფართობითი სიხშირე იქნება $P^*(N)$ – *საიმედობის შეფასება.*
8. ბ) ამოცანაში N ჩატარებული სიმულაციიდან დაითვლება t-ს მათემატიკური ლოდინის შეფასებაც.

§7.4. ზოგიერთი მასობრივი მომსახურეობის სისტემის მოდელირება სიმულაციით

7.4.1. სამ არხიანი (კვანძიანი) მასობრივი მომსახურეობის სისტემაში შემოდის მოთხოვნების (აქ შეკვეთები) პუასონის მარტივი ტიპის ნაკადი. მომდევნო შეკვეთებს შორის დრო განაწილებულია ექსპონენციალურად
$$f(\tau) = \begin{cases} 5e^{-5\tau}, & \tau > 0, \\ 0, & \tau \leq 0 \end{cases} \qquad (\lambda = 5)$$
განაწილების სიმკვრივით. ყოველი შეკვეთის მომსახურეობის დრო მუდმივია და 5 წთ-ის ტოლია. თუ ახალი შეკვეთის მიღების მომენტში სამივე მომსახურეობის არხი დაკავებულია, მაშინ შეკვეთაზე უარი კეთდება (რიგი გამორიცხულია). შეადგინე C-პროგრამა და

სიმულაციით შეაფასე მასობრივი მომსახურეობის სისტემის მიერ 4 წთ-ის განამვლობაში მომსახურეობაზე მიღებული შეკვეთების საშუალო რაოდენობა.

შენიშვნა: გთავაზობთ სქემა–სიმულაციას, რომელიც დაგეხმარებათ პროგრამის შედგენასა და ამოცანის გადაწყვეტაში.

ვთქვათ $T_1 = 0$ მომენტი პირველი შეკვეთის მომენტია. შეკვეთას მიიღებს I არხი და მოემსახურება მას. I შეკვეთის მომსახურეობის დროა $T_1 + 0{,}5 = 0 + 0{,}5 = 0{,}5$. მომსახურეობაზე მიღებული შეკვეთების მთვლელი 1-ით გაიზრდება (თავიდან იგი 0-ის ტოლი იყო).

შემდეგი შეკვეთის მიღების მომენტებს დავითვლით

$$T_i = T_{i-1} + \tau_i$$

ფორმულით, სადაც τ_i არის $(i-1)$ -დან i -ურ შეკვეთამდე მიღების დროის ხანგრძლივობა. მას მივიღებთ სიმულაციით, რომელიც გათამაშდება ფორმულით

$$\tau_i = -\frac{1}{\lambda}\ln\gamma_i = -0{,}2\ln\gamma_i \ ,$$

სადაც $\{\gamma_i\}_{i\geq 1}$ შემთხვევითი რიცხვების მიმდევრობაა, რომელიც ქვემოთ მოყვანილ ცხრილშია მოცემული.

აქ წარმოდგენილი მასობრივი მომსახურეობის სისტემის მუშაობის სიმულაცია "ხელით" ჩატარებულია შემდეგი სქემა-ცხრილის სახით:

i – შეკვეთის ნომერი	γ_i შემთხვევითი რიცხვი	$\ln\gamma_i$	მომდევნო შეკვეთებს შორის დროის ხანგრძლივობა t_i=-0,2 $\ln\gamma_i$	i-ური შეკვეთის მიღების მომენტი: $T_i = T_{i-1} + \tau_i$	$T_i + 0{,}5$ დროის მომენტი – არხის მიერ შეკვეთაზე მომსახურეობის დასრულების დროის მომენტი			მთვლელი	
					არხი 1	არხი 2	არხი 3	შეკვეთების მიღების	შეკვეთების უარის
1				0	0.500			1	
2	0.10	2.30	0.460	0.460		0.960		1	
3	0.09	2.41	0.482	0.942	1.442			1	
4	0.73	0.32	0.064	1.006		1.506		1	
5	0.25	1.39	0.278	1.284			1.784	1	
6	0.33	1.11	0.222	1.506	2.006			1	
7	0.76	0.27	0.054	1.560		2.060		1	
8	0.52	0.65	0.130	1.690					1
9	0.01	4.60	0.920	2.610	3.110			1	
10	0.35	1.05	0.210	2.820		3.320		1	
11	0.86	0.15	0.030	2.850			3.350	1	
12	0.34	1.08	0.216	3.066					1
13	0.67	0.40	0.080	3.146	3.646			1	
14	0.35	1.05	0.210	3.356		3.856		1	
15	0.48	0.73	0.146	3.502			4.002	1	
16	0.76	0.27	0.054	3.555					1
17	0.80	0.22	0.044	3.600					1
18	0.95	0.05	0.010	3.610					1

19	0.90	0.10	0.020	3.630				1	
20	0.91	0.09	0.018	3.648	4.148			1	
21	0.17	1.77	0.354	4.002 >4 გაჩერება				ჯამი=15	5

დასკვნა ასეთია: 4 წთ-ში მმს-ში შემოვიდა 20 შეკვეთა. აქედან 15 მიღებულია მომსახურეობაზე, ხოლო 5-ს უარი ეთქვა.

საჭიროა ასეთი სიმულაციის რამოდენიმეჯერ სერიების ჩატარება და მათგან საშუალოს გამოთვლა. შეადგინე C-პროგრამა და საშუალოს გრაფიკი წარმოადგინე, როგორც სერიების ფუნქცია $(N = 20, 40, 60)$.

7.4.2. სამ არხიან მასობრივი მომსახურეობის სისტემაში შემოდის შეკვეთების პუასონის მარტივი ტიპის ნაკადი. მომდევნო შეკვეთებს შორის დრო განაწილებულია ექსპონენციალურად

$$f(\tau) = \begin{cases} 4e^{-4\tau}, & \tau > 0, \\ 0, & \tau \le 0 \end{cases}$$

სიმკვრივით. ყოველი შეკვეთის მომსაურეობის დრო მუდმივია და 1 წთ-ის ტოლია. თუ ახალი შეკვეთის მიღების მომენტში სამივე მომსახურეობის არხი დაკავებულია, მაშინ შეკვეთის მიღებაზე უარი კეთდება. შეადგინე C-პროგრამა და სიმულაციით შეაფასე მმს-ის მიერ 5 წთ-ის განმავლობაში მომსახურეობაზე მიღებული შეკვეთების საშუალო რაოდენობა. ამოხსნის პირობები აიღეთ 7.4.1 ამოცანიდან.

7.4.3. ერთ არხიანი მასობრივი მომსახურეობის სისტემაში შემოდის შეკვეთების პუასონის მარტივი ტიპის ნაკადი მომდევნო შეკვეთებს შორის დრო შემთხვევითია და განაწილებულია ექსპონენციალურად

$$f(\tau) = \begin{cases} 0{,}8e^{-0{,}8\tau}, & \tau > 0, \\ 0, & \tau \le 0 \end{cases}$$

სიმკვრივით. თუ ახალი შეკვეთის მიღების მომენტში მომსახურეობის არხი დაკავებულია, მაშინ შეკვეთის მიღებაზე უარი კეთდება. შეკვეთაზე მომსახურეობის დროც შემთხვევითია და განაწილებულია ექსპონენციალურად

$$f_1(t) = \begin{cases} 1{,}5e^{-1{,}5t}, & t > 0, \\ 0, & t \le 0 \end{cases}$$

სიმკვრივით. შეადგინე C-პროგრამა და სიმულაციით 30 წთ-ის განმავლობაში იპოვე:

ა) მომსახურეობაზე მიღებული შეკვეთების საშუალო რიცხვი;
ბ) ერთი შეკვეთის მომსახურეობის საშუალო დრო;
გ) მომსახურეობის ალბათობა;
დ) ა), ბ) და გ) ამოცანებისთვის გრაფიკულად წარმოადგინე შესაფასებელი სიდიდეების, როგორც სერიების რიცხვის ფუნქცია $(N = 10, 20, 30, 40)$.

მითითება: გთავაზობთ ერთი სერიის სქემა-სიმულაციას, რომელიც დაგეხმარებათ პროგრამის შედგენასა და ამოცანის გადაწყვეტაში.

მომდევნო შეკვეთებს შორის დრო გათამაშდება ფორმულით

$$\tau_i = -(1/0.8)\ln(\gamma_i) = -1.25\ln\gamma_i .$$

γ_i შემთხვევითი რიცხვები ავიღოთ ქვემოთ მოყვანილი 7.4.3.ა ცხრილიდან.

რადგან შეკვეთაზე მომსახურეობის დრო განაწილებულია ექსპონენციალურად, ამიტომ ის გათამაშდება ფორმულით

$$t_i = -(1/2.5)\ln \gamma_i' = -0.67 \ln \gamma_i' .$$

γ_i' შემთხვევითი რიცხვები ავიღოთ ქვემოთ მოყვანილი 7.4.3.ბ ცხრილიდან.

ვთქვათ $T_1 = 0$ პირველი შეკვეთის მოსვლის მომენტია. $\gamma_i' = 0.10$ შემთხვევითი რიცხვით გათამაშდება პირველი შეკვეთის მომსახურეობის დრო (წთ-ებში)

$$t_1 = -0.67 \ln 0.10 \approx 1.54 .$$

პირველ შეკვეთაზე მომსახურეობის დასრულების მომენტია

$$T_1 = 1.54 = 0 + 1.54 .$$

შეკვეთებზე მომსახურეობის მთვლელი ერთით გაიზრდება (თავიდან იგი 0–ის ტოლია). $\gamma_2 = 0.69$ შემთხვევითი რიცხვით გათამაშდება პირველი და მეორე შეკვეთებს შორის დრო

$$\tau_2 = -1.25 \ln \gamma_2 = 0.46 .$$

პირველი შეკვეთა შემოვიდა $T_1 = 0$ მომენტში. მეორე შეკვეთა შესაბამისად შემოსულია $T_2 = T_1 + T_2 = 0 + 0.46 = 0.46$ მომენტში. ამ მომენტში არხი დაკავებულია შეკვეთაზე მომსახურეობით $(0.46 < 1.54)$. ამიტომ მეორე შეკვეთამ მომსახურეობაზე უარი მიიღო. შეკვეთებზე უარის მთვლელი ერთით გაიზრდება (თავიდან იგი 0–ის ტოლია).

$\gamma_3 = 0.07$ შემთხვევითი რიცხვით გათამაშდება მეორე და მესამე შეკვეთებს შორის დრო

$$\tau_3 = -1.25 \ln 0.07 = 3.32 .$$

მეორე შეკვეთა შემოვიდა $T_2 = 0.46$ მომენტში. მაშინ მესამე შეკვეთა შემოსულა $T_3 = T_2 + 3.32 = 0.46 + 3.32 = 3.78$ მომენტში. ამ მომენტში არხი მომსახურეობაზე თავისუფალია $(1.54 < 3.78)$. ამიტომ ის მოემსახურება მესამე შეკვეთას. შეკვეთაზე მომსახურეობის მთვლელი ერთით გაიზრდება.

ცხრილი 7.4.3.ა

შეკვეთის ნომერი i	შემთხვევითი რიცხვი γ_i	$-\ln \gamma_i$	მომდევნო შეკვეთებს შორის დრო $\tau_i = -1.25 \ln \gamma_i$	შეკვეთის მიღების მომენტი $T_i = T_{i-1} + \tau_i$
1				0
2	0.67	0.37	0.46	0.46
3	0.07	2.66	3.32	3.78
4	0.49	0.71	0.89	4.67
5	0.41	0.89	1.11	5.78
6	0.38	0.97	1.21	6.99
7	0.87	0.14	0.18	7.17
8	0.63	0.46	0.58	7.75
9	0.79	0.24	0.30	8.05
10	0.19	1.66	2.08	10.13
11	0.76	1.27	0.34	10.47
12	0.35	1.05	1.31	11.78
13	0.58	0.54	0.68	12.46
14	0.40	0.92	1.15	13.61
15	0.44	0.82	1.02	14.64
16	0.01	4.60	5.75	20.38
17	0.10	2.30	2.88	23.26
18	0.51	0.67	0.84	24.10

19	0.82	0.20	0.25	24.35
20	0.16	1.83	2.29	26.64
21	0.15	1.90	2.38	29.02
22	0.48	0.73	0.91	29.93
23	0.32	1.14	1.42	31.35 გაჩერება

ცხრილი 7.4.3.ბ

შეკვეთის ნომერი i	γ_i' შემთხვევითი რიცხვი	$\ln \gamma_i'$	შეკვეთის მომსახურეობის დრო $t_i = -0.67 \ln \gamma_i'$	მომენტი			მთვლელი	
				შეკვეთის მიღების	შეკვეთაზე მომსახურეობის დაწყების	შეკვეთაზე მომსახურეობის დასრულების	შეკვეთის მომსახურეობაზე	შეკვეთის უარზე
1	0.10	2.30	1.54	0	0	1.54	1	
2				0.46				1
3	0.09	2.41	1.61	3.78	3.78	5.39	1	
4				4.67				1
5	0.73	0.32	0.21	5.78	5.78	5.99	1	
6	0.25	1.39	0.93	6.99	6.99	7.92	1	
7				7.17				1
8				7.75				1
9	0.33	1.11	0.74	8.05	8.05	8.79	1	
10	0.76	0.27	0.18	10.13	10.13	10.31	1	
11	0.52	0.65	0.44	10.47	10.47	10.91	1	
12	0.01	4.60	3.08	11.78	11.78	14.86	1	
13				12.46				1
14				13.61				1
15				14.63				1
16	0.35	1.05	0.70	20.38	20.38	21.08	1	
17	0.86	0.15	0.10	23.26	23.26	23.36	1	
18	0.34	1.08	0.72	24.10	24.10	24.82	1	
19				24.35				1
20	0.67	0.40	0.27	26.64	26.64	26.91	1	
21	0.35	1.05	0.70	29.02	29.02	29.72	1	
22	0.48	0.73	0.49	29.93	29.93		1	
$\sum$			11.71				14	8

“ხელით” ჩატარებული სიმულაციით მიღებული გამოთვლები წარმოდგენილია 7.4.3.ა და 7.4.3.ბ ცხრილებში. სიმულაცია დასრულდება მაშინ, როდესაც i-ური შეკვეთის მიღების მომენტი $T_i \geq 30$. როგორც 7.4.3.ა ცხრილიდან ჩანს 23–ე შეკვეთა ასრულებს ექსპერიმენტს – ერთ სერიას.

ამ სერიაში, 30 წთ-ის განმავლობაში მიღებულია 14 შეკვეთა მომსახურეობაზე, ხოლო 8-ს უარი ეთქვა. სისტემამ მომასხურეობაზე სულ დახარჯა 11,71 წთ. შეკვეთის მიღების ალბათობის შეფასებაა $\hat{P} = \frac{14}{22} = \frac{7}{11}$, ხოლო უარისა კი $\frac{8}{22} = \frac{4}{11}$. საჭიროა ანალოგიურად სიმულაციები ჩატარდეს N სერიაში $(N = 10, 20, \ldots)$ და დაითვალოს შესაბამისი საშუალოები.

მოვიყვანოთ პირობითი ცხრილი (ცხრილი 7.4.3.გ), თუ სიმულაციათა რიცხვია $N = 6$. ამ ცხრილით შევაფასებთ იმ სიდიდეებს, რომელიც წარმოდგენილია ამოცანაში:

ა) 30წთ-ის განმავლობაში მომსახურეობაზე მიღებული შეკვეთების საშუალო რიცხვი არის: $93/6 = 15{,}5$;

ბ) შეკვეთაზე მომსახურეობის საშუალო დროა: $4.49/6 = 0.748$;

გ) შეკვეთაზე მომსახურეობის ალბათობის შეფასებაა: $3{,}973/6 = 0.662$.

დასკვნა: შეკვეთების დაახლოებით 66% მიიღება მომსახურეობაზე;

დ) ეს ამოცანა აზუსტებს წინა ამოცანის შეფასებებს, როცა სერიათა რიცხვი იზრდება.

ცხრილი 7.4.3.გ

j სერიის ნომერი	N_j – სერიაში შეკვეთების რიცხვი	N_j^+ – სერიაში მომსახურეობაზე მიღებული შეკვეთების რიცხვი	t_j – სერიაში შეკვეთებზე მომსახურეობის დრო	t_j/N_j^+ – სერიაში ერთ შეკვეთაზე საშუალო მომსახურეობის დრო	N_j^+/N_j – სერიაში მომსახურეობაზე შეკვეთის მიღების სიხშირე	$1\text{-}N_j^+/N_j$ – სერიაში მომსახურეობაზე უარის თქმის სიხშირე
1	22	13	11.71	0.90	0.591	0.409
2	25	17	8.80	0.52	0.680	0.320
3	24	16	13/46	0.84	0.607	0.333
4	22	15	12.19	0.81	0.682	0.318
5	20	13	11.99	0.92	0.650	0.350
6	27	19	3.57	0.50	0.704	0.296
$\sum$	140	93		4.49	3.973	

7.4.4. იგივე ამოცანა, რაც 7.4.3, ოღონდ

$$f(\tau) = \begin{cases} 0.5e^{-0.5\tau}, & \tau > 0, \\ 0, & \tau \le 0, \end{cases}$$

$$f_1(t) = \begin{cases} 2e^{-2t}, & t > 0, \\ 0, & t \le 0 \end{cases}.$$

სისტემაზე სიმულაციის ჩატარების დროა 20 წთ.

7.4.5. იგივე ამოცანა, რაც 7.4.3, ოღონდ მმს შედგება 3 დამოუკიდებელი არხისგან. თითოეული არხის შეკვეთაზე მომსახურეობის დრო შემთხვევითია და განაწილებულია ექსპონენციალურად:

I არხი: $f_1(t)=\begin{cases}1{,}2e^{-1,2t}, & t>0,\\ 0, & t\le 0;\end{cases}$

II არხი: $f_2(t)=\begin{cases}1{,}4e^{-1,4t}, & t>0,\\ 0, & t\le 0;\end{cases}$

III არხი: $f_3(t)=\begin{cases}1{,}6e^{-1,6t}, & t>0,\\ 0, & t\le 0.\end{cases}$

მომდევნო შეკვეთებს შორის დრო განაწილებულია ექსპონენციალურად

$$f(\tau)=\begin{cases}0.8e^{-0.8\tau}, & \tau>0,\\ 0, & \tau\le 0.\end{cases}$$

სისტემის სიმულაციის დრო შეადგენს 30 წთ–ს.

7.4.6.- 7.4.13. თავი 5-ის 5.6.7* და 5.6.14* ამოცანები.

§7.5. დაზღვევის მოდელი და მისი სიმულაცია

წარმოგიდგენთ სადაზღვევო კომპანიის მოდელს, სადაც ირკვევა, რომ სადაზღვევო სისტემის ფუნქციონირების ანალიზურად *წინასწარი შეფასება პრაქტიკულად შეუძლებელია* და ერთადერთ გზად *სიმულაცია* რჩება.

მოდელის აღწერა: ვთქვათ t დრო ღებულობს მთელ მნიშნელობებს $1, 2, 3, \ldots, N$, სადაც ბიჯი შეიძლება წარმოვიდგინოთ, როგორც მოდელის დროის ერთეული (საათი, დღე ან დღის რაღაც ნაწილი, N ასევე შეიძლება იყოს ½ წელი, 3 თვე და ა.შ.). ვთქვათ დროის ამ ერთეულში შეიძლება განხორციელდეს *ზუსტად ერთი მოთხოვნა (სადაზღვევო შემთხვევა) მოცემულ პოლისზე დაზღვევის პორტფელიდან* რაიმე $p>0$ ალბათობით. დავუშვათ, რომ *სადაზღვევო პორტფელი* შედგება M პოლისისგან (ჩვენ ვუშვებთ, რომ დაზღვევის დროის ერთეულში ერთზე მეტი მოთხოვნა არ განხორციელდება; ცხადია დროის ბიჯი უკუპროპორციული M-ის). *პოლისის მოთხოვნის თანხა* შემთხვევითია და მისი განაწილების ფუნქცია აღვნიშნოთ F-ით (თანხა რაიმე ერთეულებში იზომება: ლარი, დოლარი, . . .).

ვთქვათ პორტფელის მოქმედების პერიოდში (N ერთ. დრო) სადაზღვევო კომპანია მიიღებს რაიმე $\hat{\Pi}$ *შემოსავალს*, რომელიც წარმოადგენს დაზღვეულ პირთა მიერ კომპანიაში შენატანების აკუმულირებელ ჯამს N დროის განმავლობაში. ვიგულისხმოთ, რომ $\hat{\Pi}$ მუდმივი სიდიდეა. ვთქვათ სადაზღვევო კომპანიის *სარეზერვო ფონდია* U, რომელიც აუცილებელია პორტფელის შექმნის დასაწყისში $(t=0)$.

ამოცანა მდგომარეობს შემდეგში: პორტფელის მოქმედების ვადის ბოლოსთვის (N ერთ. დრო) შევაფასოთ (წინასწარ) სადაზღვევო კომპანიის *გაკოტრების ალბათობა.*

მითითება. ვთქვათ T არის პორტფელის მოქმედების პერიოდში დაზღვეულთა მიერ სადაზღვევო შემთხვევისათვის *კომპანიიდან მიღებულ თანხათა ჯამი* (კომპანიის მიერ პოლისებზე გადახდილი თანხა). მაშინ შესაფასებელი ალბათობაა

$$\Pi(U+\hat{\Pi}<T); \qquad (7.10)$$

ე.ი. ალბათობა იმისა, რომ გადახდილი თანხა აჭარბებს სადაზღვევო კომპანიის ფინანსურ რესურსებს (კომპანია გაკოტრებულია).

გაკოტრების ალბათობის *მიახლოებითი შეფასებისთვის* შეიძლება ჩავატაროთ შემდეგი ანალიზი:

T შეიძლება წარმოვადგინოთ, როგორც ჯამი

$$T = c_1 + c_2 + \cdots + c_r \,, \tag{7.11}$$

სადაც r მოთხოვნათა რიცხვია, $c_1\, c_2, \ldots$ ინდივიდუალური გადახდილი თანხებია პოლისის მოთხოვნაზე. დავუშვათ, რომ r შემთხვევითი სიდიდე განაწილებულია ბინომიალური კანონით

$$\Pi(r = k) = C_N^k\, p^k (1-P)^{N-k}, \qquad k = 0, 1, \ldots, N \tag{7.12}$$

(c_i მოთხოვნილი თანხაა, რომლის განაწილების კანონია F; ისინი ერთნაირად განაწილებული, დამოუკიდებელი შემთხვევითი სიდიდეებია). ვთქვათ F კანონიდან გამოვითვალეთ პოლისის მოთხოვნაზე გადახდის მათემატიკური ლოდინი $\mathrm{E}c_i = m$ და დისპერსია $Dc_i = \sigma^2$. შევნიშნოთ, რომ სრული ალბათობის ფორმულით (მათემატიკური ლოდინისთვის) ვღებულობთ:

$$\begin{aligned} \mathrm{E}(T) &= \sum_{k=0}^{N} \Pi(r = k) \cdot \mathrm{E}(T/r = k) = \sum_{k=0}^{N} \Pi(r = k) \cdot (km) = \\ &= m \sum_{k=0}^{N} k \cdot \Pi(r = k) = m \cdot \mathrm{E}(r) = m \cdot Np \,, \end{aligned} \tag{7.13}$$

რადგან $\mathrm{E}(T/r = k) = \mathrm{E}\left(\sum_{i=1}^{k} c_i\right) = \sum_{i=1}^{k} \mathrm{E}c_i = k \cdot m$, ხოლო r – ბერნულის განაწილების მათემატიკური ლოდინია Np. ანალოგიურად გამოვთვალოთ:

$$\begin{aligned} \mathrm{E}(T^2) &= \sum_{k=0}^{N} \Pi(r = k) \cdot \mathrm{E}(T^2/r = k) = \sum_{k=0}^{N} \Pi(r = k) \cdot (k\sigma^2 + k^2 m^2) = \\ &= \sigma^2 \sum_{k=0}^{N} \Pi(r = k) \cdot k + m^2 \sum_{k=0}^{N} \Pi(r = k) \cdot k^2 = \sigma^2 \cdot \mathrm{E}(r) + m^2 \cdot \mathrm{E}(r^2) = \\ &= \sigma^2 \cdot Np + m^2 \left[(Np)^2 + Np(1-p) \right], \end{aligned} \tag{7.14}$$

რადგან დისპერსიის განმარტებით

$$\begin{aligned} \mathrm{E}(T^2/r = k) &= \Delta(T/r = k) + [\mathrm{E}(T/r = k)]^2 = \\ &= \Delta\left(\sum_{i=1}^{k} c_i\right) + (km)^2 = k \cdot \sigma^2 + k^2 m^2 \end{aligned}$$

და

$$\mathrm{E}r^2 = \Delta r + (\mathrm{E}r)^2 = Np(1-p) + (Np)^2 \,.$$

მაშინ გამოვთვლით T-ს უპირობო დისპერსიას:

$$\begin{aligned} \Delta T &= \mathrm{E}T^2 - (\mathrm{E}T)^2 = \sigma^2 Np + m^2 [N^2 p^2 + Np(1-p)] - m^2 N^2 p^2 = \\ &= \sigma^2 Np + m^2 Np(1-p) = Np\left\{\sigma^2 + m^2(1-p)\right\}. \end{aligned}$$

ცენტრალური ზღვრითი თეორემის ძალით, თუ N დიდია ($Np >> 1$) და r დიდია ($r >> 1$), T ასიმპტოტურად ნორმალურადაა განაწილებული (ჩვენს შემთხვევაში სადაზღვევო შემთხვევები "ხშირად" ხდება), მაშინ გაკოტრების ალბათობის აპროქსიმაცია ასეთია:

$$\begin{aligned} \Pi(\text{კომპანია გაკოტრდება}) &= \Pi(T > U + \hat{\Pi}) = 1\text{-}\Pi(T \le U + \hat{\Pi}) \approx \\ &\approx 1\text{-}\Phi\left(\frac{U + \hat{\Pi}\text{-}\mathrm{E}T}{\sqrt{\Delta T}}\right), \end{aligned}$$

საიდანაც

$$\Pi(\text{კომპანია გაკოტრდება}) \approx 1\text{-}\Phi\left(\frac{U+\hat{\Pi}\text{-}Npm}{\sqrt{Np(\sigma^2+m^2(1-p)}}\right), \qquad (7.15)$$

სადაც Φ ლაპლასის ფუნქციაა (მისი ცხრილები ჩვენთვის ცნობილია სტატისტიკის კურსიდან).

(7.15) ფორმულით სარგებლობა შეიძლება თუ m და σ^2 ჩვენთვის ცნობილია. მათი შეფასება სტატისტიკური მეთოდებით შესაძლებელია *გასული პორტფელების მონაცემების* საშუალებით, თუკი N დიდია და r შემთხვევითი რიცხვიც დიდია. წინააღმდეგ შემთხვევაში ამ ფორმულით *სარგებლობის საიმედობა დაბალია* [4].

ასეთ შემთხვევაში და უფრო რთული სადაზღვევო *მოდელებით* სარგებლობისას (როცა ელემენტარულად (7.15) ფორმულის მსგავსი შეფასება ვერ კეთდება) მიმართავენ სიმულაციას.

სიმულაცია ასევე ერთადერთი მეთოდია, როდესაც სადაზღვევო კომპანია *ნერგავს თავის ბიზნესში ახალ მოდელებს* და გასული პორტფელების მონაცემებით სარგებლობა უკვე შეუძლებელი ხდება (m და σ^2–ის შეფასება ვერ ხერხდება). ასეთ შემთხვევაში $M, U, \hat{\Pi}, N, p$ პარამეტრების ვარირებით, r-ისა და c_i-ების სიმულირებით იქმნება "*სადაზღვევო მოდელის სცენარები*", რომელიც იკვლევს კომპანიის ამოცანებს ახალი ტიპის პოლისების შედგენაში. შემდეგ კი ირჩევა გარკვეული ოპტიმალური სცენარი, რომელიც უკვე ჩამოაყალიბებს სადაზღვევო კომპანიის პორტფელის სტრუქტურას.

განვიხილოთ კონკრეტული მაგალითები:

სიმულაციით შეაფასეთ გაკოტრების ალბათობა და შეადარეთ იგი (7.15) აპროქსიმაციის ფორმულით გამოთვლილ ალბათობას.

შენიშვნა: შეადგინე C-პროგრამა, რომლის შემავალი პარამეტრები იქნება N, p, m, σ^2, $U, \hat{\Pi}$; c_i – განაწილებული იქნება ნორმალურად m, σ^2 პარამეტრებით: $c_i \sim N/(m, \sigma^2)$.

სიმულაციის წესი: ჯერ გათამაშდება r, ანუ პოლისზე მოთხოვნათა რიცხვები

$$P_k = P(r=k) = C_N^k p^k (1-p)^{N-k}, \quad k = 0, 1, \ldots, N,$$

კანონით (0,1). ინტერვალი დაიყოფა $N+1$ თანაუკვეთ ინტერვალებად: $\{\Delta_k\}_{k=0}^N$, Δ_k-ს სიგრძეა p_k: $\Delta_0 = (0, p_0]$, $\Delta_1 = (p_0; p_0 + p_1], \ldots, \Delta_N = (p_0 + \cdots + p_{N-1}, 1)$. ამ ინტერვალების აგება ადვილია, თუ ვისარგებლებთ რეკურენტული ფორმულით:

$$p_{k+1} = p_k \cdot r(k), \quad k = 0, 1, \ldots, N-1; \quad p_0 = (1-p)^N,$$

სადაც

$$r(k) = \frac{p_{k+1}}{p_k} = \frac{N-k}{k+1} \cdot \frac{p}{1-p}.$$

თუ γ შემთხვევითი რიცხვი ჩავარდა Δ_j ინტერვალში, მაშინ პოლისზე მოთხოვნათა რიცხვია $r = j$.

შემდეგ გათამაშდება j რაოდენობა დამოუკიდებელი ნორმალური შემთხვევითი რიცხვი m და σ^2 პარამეტრებით: $c_1, c_2, \ldots, c_j$, რაც მოგვცემს სიმულირებით მიღებულ T სიდიდეს:

$$T = c_1 + c_2 + \cdots + c_j .$$

თუ $T > U + \hat{\Pi}$, მაშინ კომპანია "გაკოტრებულია",

თუ $T \le U + \hat{\Pi}$, მაშინ კომპანია "არაა გაკოტრებული".

ეს სიმულაცია ჩაატარეთ L რაოდენობა სერიით და P_L გაკოტრების ფარდობითი სიხშირე იქნება გაკოტრების ალბათობის შეფასება.

მაგალითები (შეავსეთ ცხრილი)

i – მაგალითის ნომერი	Π – ლარებში	U – ლარებში	P – მოთხოვნის ალბათობა ერთეულოვან დროში	m – მოთხოვნის საშუალო	σ^2 – მოთხოვნის დისპერსია	N – პორტფელის ხანგრძლივობა (დღე)	გაკოტრების ალბათობის აპროქსიმაცია და მისი შედარება P_L – გაკოტრების ალბათობის სიმულაციის გზით მიღებულ შეფასებასთან				
							L სერიები	P_L	$\Pi_{\text{აპროქ}}$	$\lvert \Pi_L - \Pi_{\text{აპროქ}} \rvert$	$\lvert P_L - P_{2L} \rvert$
7.5.1	10000	25000	0.4	20	4	364	20				
7.5.2	50000	75000	0.2	100	9	364	20				
7.5.3	10^6	$3{\cdot}10^6$	0.02	200	16	5·364	100				
7.5.4	10^6	$4{\cdot}10^6$	0.01	500	64	5·364	100				
7.5.5	10^6	$3{\cdot}10^6$	0.01	400	81	5·364	100				
7.5.6	$2{\cdot}10^6$	$4{\cdot}10^6$	0.04	200	16	10·364	100				

შენიშვნა: ყოველი მაგალითის შემთხვევაში ჩაატარეთ ანალიზი გაკოტრების ალბათობის შეფასებიდან.

§7.6. ძიების ამოცანა და მისი სიმულაცია

პრაქტიკაში ხშირია შემთხვევა, როდესაც რაიმე საძიებელი ობიექტი შეიძლება იმყოფებოდეს წინასწარ ცნობილი N რაოდენობა ობიექტებიდან ერთ-ერთში. საძიებო ობიექტის ოპტიმალურად მოკლე დროში აღმოჩენისთვის მნიშვნელოვანია ეფექტური ალგორითმების შედგენა, როდესაც N საკმარისად დიდია! თუ ამოცანა დეტერმინისტულია, მაშინ არსებობს ოპტიმალური ძიების უამრავი ალგორითმი, როდესაც აპრიორული ინფორმაციის შემთხვევაში აპოსტერიული, ახალი ინფორმაცია ძიებაზე მიიღება დეტერმინისტულად.

ჩვენ აქ განვიხილავთ სტოქასტურ საძიებო მოდელს (აპრიორული ინფორმაცია შემთხვევითი ფაქტორია).

მოდელის აღწერა: ამოსაცნობი ობიექტი მოთავსებულია ერთ-ერთ ობიექტში ნომრებით $1, 2, \ldots, N$. რომ აღმოვაჩინოთ, თუ რომელ ობიექტში იმყოფება საძიებელი ობიექტი, ჩვენ უფლება გვაქვს დავსვათ შემდეგი ტიპის შეკითხვა. იმყოფება თუ არა საძიებელი ობიექტი ერთ-ერთ $(l+1)$ რაოდენობა მეზობელ ობიექტებში ნომრებით:

$$a, a+1, a+2, \ldots, b. \tag{7.16}$$

პასუხი, რომელსაც ჩვენ მივიღებთ არ იქნება სანდო მთლიანობაში და სამართლიანი იქნება გარკვეულ $W(l)$ ალბათობით $(l = 0, 1, \ldots, N-1)$, $l = b - a$. თუ ჩვენ ვკითხულობთ მხოლოდ ერთ ობიექტში ძიებაზე, მაშინ $W(0)$ ალბათობა ახლოს უნდა იყოს 1-თან. l-ის ზრდასთან ერთად $W(l)$ კლებადი იქნება 0,5–კენ, სადღაც $\left[\frac{l}{2}\right]$ მთელ ნაწილამდე, შემდეგ ისევ ზრდადი გახდება, რადგან

$$W(l) = W(N-1-l), \qquad l = 0, 1, 2, \ldots, N-1. \tag{7.17}$$

სიმეტრიის პრინციპი გასაგებია, რადგან შეკითხვები l და $N-l-1$ ობიექტებში ამოსაცნობი ობიექტის არსებობაზე ტოლფას ინფორმაციას იძლევიან.

დაშვებულია, რომ ძიების დაწყების წინ ჩვენ გაგვაჩნია გარკვეული საწყისი, აპრიორული განუზღვრელი ინფორმაცია, თუ სად იმყოფება ამოსაცნობი ობიექტი. ამ ინფორმაციას ფორმალიზებას ვუკეთებთ გარკვეული აპრიორული

$$P = (p_1,\ p_2, \ldots,\ p_N), \qquad \sum_{i=1}^{N} p_i = 1, \tag{7.18}$$

განაწილების სახით, სადაც $p_i > 0$ არის ალბათობა იმისა, რომ ამოსაცნობი ობიექტი იმყოფება i-ურ ობიექტში.

დავუშვათ, რომ შეკითხვისთვის ჩვენ ავარჩიეთ $a \le b$ $(l = b - a)$ წყვილი და ვღებულობთ ერთ-ერთ პასუხს $W(l)$ ალბათობით:

$$\begin{array}{c} \text{იმყოფება} \leftrightarrow \text{“პასუხი 1”} \\ \text{არ იმყოფება} \leftrightarrow \text{“პასუხი 0”} . \end{array} \tag{7.19}$$

თითოეული პასუხი რა თქმა უნდა გადააფასებს ინფორმაციას, აპრიორულ განაწილებას, თუ რა ალბათობით იმყოფება ამოსაცნობი ობიექტი მოცემულ $1, 2, \ldots, N$ ობიექტებში. ბაიესის ფორმულით ვითვლით

$$Q = (q_1,\ q_2, \ldots,\ q_N), \qquad \sum_{i=1}^{N} q_i = 1, \tag{7.20}$$

აპოსტერიულ განაწილებას. მაგრამ განსახილველია 2 შემთხვევა:
1 შემთხვევა – “პასუხი 1”. მაშინ ვღებულობთ:

$$\begin{aligned} q_i &= \Pi(\text{საძიებელი ობიექტი არის } i\text{-ურ ობიექტში} \ / \ \text{"პასუხი 1"}) = \\ &= \frac{\Pi(\text{საძიებელი ობიექტი არის } i\text{-ურ ობიექტშია და "პასუხი 1"})}{\Pi(\text{"პასუხი 1"})} = \\ &= \frac{p_i \cdot \Pi(\text{"პასუხი 1"} \ / \ \text{საძიებელი ობიექტი } i\text{-ურ ობიექტშია})}{\Pi(\text{"პასუხი 1"})} . \end{aligned} \tag{7.21}$$

ჩვენ გვაქვს, თუ $j \in [a;b]$, მაშინ

$$\Pi(\text{"პასუხი 1"} \ / \ \text{საძ. ობიექტი } j\text{-ურ ობიექტშია}) = \Pi(\text{"პასუხი 1" სწორია}) = W(l), \tag{7.22}$$

და თუ $j \notin [a;b]$, მაშინ

$$\Pi(\text{"პასუხი 1"} \ / \ \text{საძ. ობიექტი } j\text{-ურ ობიექტშია}) = \Pi(\text{"პასუხი 1" მცდარია}) = 1 - W(l), \tag{7.22´}$$

მაშინ ამ ფორმულებით გამოვითვლით Π (“პასუხი 1”) ალბათობას სრული ალბათობის ფორმულით:

$$\Pi(\text{"პასუხი 1"}) = \sum_{j=1}^{n} p_j \cdot \Pi(\text{"პასუხი 1"} \ / \ \text{საძ. ობიექტი } j\text{-ურ ობიექტშია})$$

ანუ

$$\Pi(\text{"პასუხი 1"}) = \sum_{j=1}^{n} p_j \cdot \left(\chi_{[a;b]}(j) \cdot W(l) + \chi_{\overline{[a;b]}}(j) \cdot (1 - W(l))\right),$$

სადაც χ სიმრავლის მახასიათებელი ფუნქციაა. მაშინ ვღებულობთ:

$$q_i = \frac{p_i \left(\chi_{[a;b]}(i) \cdot W(l) + \chi_{\overline{[a;b]}}(i) \cdot (1 - W(l))\right)}{\sum_{j=1}^{n} p_j \cdot \left(\chi_{[a;b]}(j) \cdot W(l) + \chi_{\overline{[a;b]}}(j) \cdot (1 - W(l))\right)}, \quad i = 1, 2, \ldots, N, \tag{7.23}$$

ასე, რომ ავაგეთ $Q = (q_1, q_2, \ldots, q_N)$ განაწილება, თუ მივიღებდით “პასუხი 1”–ს.

ანალოგიურად “პასუხი 0”–ის მიღების შემთხვევაში გვექნება:

$$q_i = \frac{p_i\left(\chi_{[a;b]}(i)\,(1-W(l)) + \chi_{\overline{[a;b]}}(i)\cdot W(l)\right)}{\sum_{j=1}^{n} p_j \cdot \left(\chi_{[a;b]}(j)(1-W(l)) + \chi_{\overline{[a;b]}}(j)\cdot W(l)\right)}, \quad i = 1, 2, \ldots, N, \tag{7.24}$$

შემდეგი ნაბიჯი ასეთია: თუ აღმოჩნდა, რომ Q განაწილება თავმოყრილია ვთქვათ ძირითადად რაიმე i_0 ობიექტში, მაშინ ჩვენ შეიძლება ჩამოვაყალიბოთ გადაწყვეტილება: "საძიებელი ობიექტი i_0 ობიექტშია". თუ აღმოჩნდა, რომ აპოსტერიული განაწილება თავმოყრილია ძირითადად ორ რაიმე i_1 და i_2 ობიექტში, მაშინაც შეგვიძლია ჩამოვაყალიბოთ გადაწყვეტილება: "საძიებელი ობიექტი ან i_1 ან i_2 ობიექტია" და ა.შ.

რას გვინდა მივაღწიოთ? ჩვენი მიზანია შეკითხვაზე პასუხის მიღების შემდეგ გავაკეთოთ რამდენადაც შესაძლებელია ზუსტი *განმარტება*: "საძიებელი ობიექტი იმყოფება რაც შეიძლება მცირე რაოდენობა ელემენტების $\{i_2, i_2, \ldots\}$ სიმრავლეში". მეორე მხრივ ეს ნიშნავს, რომ a და b -ს ხელახალი არჩევით (ესეც ამოცანაა, თუ როგორ ავირჩიოთ a და b) აპოსტერული განაწილება ჩავთვალოთ უკვე აპრიორულად და ბაიესის ფორმულის ხელახალი გამოყენებით გადავათამაშოთ აპოსტერიული განაწილება, რამეთუ Q განაწილება *უფრო ნაკლებ რაოდენობა ობიექტებში იქნება თავმოყრილი* (კონცენტრირებული). განაწილების კონცენტრაციის *ზომად ავიღოთ*:

$$CRIT = \sum_{i=1}^{N} q_i^2, \tag{7.25}$$

ან

$$ENTRO = -\sum_{i=1}^{N} q_i \ln q_i. \tag{7.26}$$

დავაკონკრეტოთ: შევქმნათ C-პროგრამა, სადაც a და b რიცხვების შერჩევის შესაძლებლობა იქნება. განსხვავებული $(a;b)$ წყვილებისთვის დავთვალოთ ან (7.25) ან (7.26)-ით არჩეული კრიტერიუმები და ავირჩიოთ ისეთი $(a;b)$, რომლისთვისაც კრიტერიუმის მნიშვნელობა *უდიდესი იქნება*. მივიღოთ გადაწყვეტილება ძიებაზე. a და b –ს არჩევანი დიალოგშია გასაკეთებელი და ის დამოკიდებულია აპრიორულ

$$(p_1, p_2, \ldots, p_n)$$

განაწილებაზე. უნდა ავირჩიოთ ისეთი a და b , რომ $[a;b]$-ში უფრო თავმოყრილი იყოს Q განაწილება.

ძიების ამოცანის სიმულაციის წესი: ცხადია აპოსტერული $Q = (q_1, q_2, \ldots, q_N)$ განაწილება ბაიესის პროცესით მოდელირების ყოველ ბიჯზე (a და b -ს არჩევა, პასუხის მიღება და Q განაწილების გამოთვლა) საჭიროა ვისარგებლოთ ან (7.23) ან (7.24) ფორმულით. ამისთვის საჭიროა შეკითხვაზე მიღებული პასუხის სიმულაცია. ანუ

$$\begin{pmatrix} \text{"პასუხი 1"} & \text{"პასუხი 0"} \\ W(l) & 1-W(l) \end{pmatrix}$$

განაწილების გათამაშება.

თუ γ შემთხვევითი რიცხვია და $0 < \gamma \le W(l)$, მაშინ გვაქვს "პასუხი 1";

თუ $W(l) < \gamma < 1$, მაშინ გვაქვს "პასუხი 0";

ამოცანები:

დინამიურად წარმოადგინე N და მაშასადამე $P_1, P_2, \ldots, P_N$ განაწილება და $W(l)$, $l = 0, 1, \ldots, N-1$, ალბათობათა ვექტორი (7.17) პირობებით. ზემოთ მოყვანილი მოდელის სიმულაციის სქემა დააპროგრამირე a და b პარამეტრების დიალოგურ რეჟიმში არჩევის

სახით. პროგრამა ჩერდება დიალოგურ რეჟიმში და ირჩევა a_0, b_0 (მაშასადამე l) და აქედან გამომდინარე ისეთი აპოსტერიული $Q = (q_1, q_2, \ldots, q_N)$ განაწილება, რომლის კრიტერიუმის მნიშვნელობა უდიდესია. კრიტერიუმის არჩევაც დიალოგში (2–დან ერთი) შეირჩევა.

ამოხსენით რიცხვითი მაგალითი:

$N = 5$ ობიექტი

ობიექტები	1	2	3	4	5
აპ/განწ: П	0,1	0,15	0,35	0,2	0,2

l	0	1	2	3	4
$W(l)$	0,9	0,7	0,6	0,7	0,9

კრიტერიუმი იყოს $CRIT = \sum_{i=1}^{s} q_i^2$. "ხელით" ძიების სიმულაცია და დიალოგური მოდელირება წარმოდგენილია ცხრილში:

ცხრილი: მოდელირების სქემა

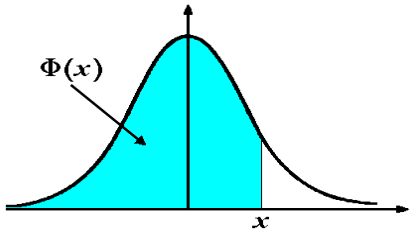
i მოდელირების ბიჯი	γ_i შემთხვევითი რიცხვი	a და b-ს არჩევა	$l=b-a$	"პასუხის" გენერირება	Q აპოსტერიული განაწილების გამოთვლა	$CRIT$ კოიტერიუმის გამოთვლა	განაწყვეტილება მოდელირების გაგრძელებაზე	განაწყვეტილება ძიებაზე	კომენტარი დიალოგისთვის
1	0.21	a=3 b=4	l =1	"3.1" იმყოფება	$q_1 = 0.058$ $q_2 = 0.087$ $q_3 = 0.471$ $q_4 = 0.269$ $q_5 = 0.115$	CRIT=0.01265	გაგრძელება	–	ა) აირჩა a=3 და b=4. იმიტომ რომ მე-3 და მე-4 ობიექტებს ყველაზე მაღალი ალბათობები აქვთ ბ) აპოსტერული განაწილება თავმოყრილია მე-3 და მე-4 ობიექტებზე

2	0.75	a=3 b=5	l =2	“3.0” არ იმყოფება	$q_1 = 0.08$ $q_2 = 0.1338$ $q_3 = 0.4333$ $q_4 = 0.2474$ $q_5 = 0.1057$	CRIT=0.029543	გაგრძელება	–	ა) აპრიორული განაწილება ახლა არის 1 ბიჯის აპოსტერული: $P_1 = 0.058,\ P_2 = 0.087,\ P_3 = 0.471,$ $P_4 = 0.269,\ \ P_5 = 0.115,$ ბ) მე-3 და მე-4 ალბათობები კვლავ გაიზარდა. გაიზარდა კრიტერიუმიც. სრული საფუძველია მივიღოთ გადაწყვეტილება მათ სასარგებლოდ. მაგრამ კიდევ გავაგრძელოთ სიმულაცია
3	0.46	a=2 b=4	l =2	“3.1” იმყოფება	$q_1 = 0.032$ $q_2 = 0.0803$ $q_3 = 0.2597$ $q_4 = 0.1484$ $q_5 = 0.0423$	CRIT=0.098812	შეწყდეს სიმულაცია	სანამ ობიექტი იმყოფება III ან IV ობიექტებში	ა) აპრიორული განაწილება ახლა არის მე-2 ბიჯის აპოსტერული: $P_1 = 0.08,\ P_2 = 0.1138,\ P_3 = 0.4331,$ $P_4 = 0.2474,\ \ P_5 = 0.1057,$ ბ) მე-3 და მე-4 ალბათობები სტა-ბილურად კვლავ აღემატება დანარ-ჩენებს. კრიტერიუმი დაგრძნობლად გაიზარდა. გვაფიქრებინებს, რომ შევწყვიტოთ მოდელირება მე-3 ან მე-4 ობიექტის სასარგებლოდ

დანართი

1. სტანდარტული ნორმალური განაწილების ფუნქციის მნიშვნელობების ცხრილი

$$\Phi(x)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{t^2}{2}}dt$$

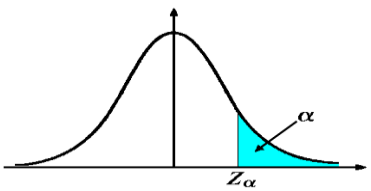


x	Φ(x)	x	Φ(x)	x	Φ(x)	x	Φ(x)	x	Φ(x)	x	Φ(x)
0.00	0.500	**0.33**	0.629	**0.66**	0.745	**0.99**	0.838	**1.32**	0.906	**1.65**	0.950
0.01	0.503	**0.34**	0.633	**0.67**	0.748	**1.00**	0.841	**1.33**	0.908	**1.66**	0.951
0.02	0.507	**0.35**	0.636	**0.68**	0.751	**1.01**	0.843	**1.34**	0.909	**1.67**	0.952
0.03	0.511	**0.36**	0.640	**0.69**	0.754	**1.02**	0.846	**1.35**	0.911	**1.68**	0.953
0.04	0.515	**0.37**	0.644	**0.70**	0.758	**1.03**	0.848	**1.36**	0.913	**1.69**	0.954
0.05	0.519	**0.38**	0.648	**0.71**	0.761	**1.04**	0.850	**1.37**	0.914	**1.70**	0.955
0.06	0.523	**0.39**	0.651	**0.72**	0.764	**1.05**	0.853	**1.38**	0.916	**1.71**	0.956
0.07	0.527	**0.40**	0.655	**0.73**	0.767	**1.06**	0.855	**1.39**	0.917	**1.72**	0.957
0.08	0.531	**0.41**	0.659	**0.74**	0.770	**1.07**	0.857	**1.40**	0.919	**1.73**	0.958
0.09	0.535	**0.42**	0.662	**0.75**	0.773	**1.08**	0.859	**1.41**	0.920	**1.74**	0.959
0.10	0.539	**0.43**	0.666	**0.76**	0.776	**1.09**	0.862	**1.42**	0.922	**1.75**	0.959
0.11	0.543	**0.44**	0.670	**0.77**	0.779	**1.10**	0.864	**1.43**	0.923	**1.76**	0.960
0.12	0.547	**0.45**	0.673	**0.78**	0.782	**1.11**	0.866	**1.44**	0.925	**1.77**	0.961
0.13	0.551	**0.46**	0.677	**0.79**	0.785	**1.12**	0.868	**1.45**	0.926	**1.78**	0.962
0.14	0.555	**0.47**	0.680	**0.80**	0.788	**1.13**	0.870	**1.46**	0.927	**1.79**	0.963
0.15	0.559	**0.48**	0.684	**0.81**	0.791	**1.14**	0.872	**1.47**	0.929	**1.80**	0.964
0.16	0.563	**0.49**	0.687	**0.82**	0.793	**1.15**	0.874	**1.48**	0.930	**1.81**	0.964
0.17	0.567	**0.50**	0.691	**0.83**	0.796	**1.16**	0.876	**1.49**	0.931	**1.82**	0.965
0.18	0.571	**0.51**	0.694	**0.84**	0.799	**1.17**	0.879	**1.50**	0.933	**1.83**	0.966
0.19	0.575	**0.52**	0.698	**0.85**	0.802	**1.18**	0.881	**1.51**	0.934	**1.84**	0.967
0.20	0.579	**0.53**	0.701	**0.86**	0.805	**1.19**	0.882	**1.52**	0.935	**1.85**	0.967
0.21	0.583	**0.54**	0.705	**0.87**	0.807	**1.20**	0.884	**1.53**	0.936	**1.86**	0.968
0.22	0.587	**0.55**	0.708	**0.88**	0.810	**1.21**	0.886	**1.54**	0.938	**1.87**	0.969
0.23	0.590	**0.56**	0.712	**0.89**	0.813	**1.22**	0.888	**1.55**	0.939	**1.88**	0.969
0.24	0.594	**0.57**	0.715	**0.90**	0.815	**1.23**	0.890	**1.56**	0.940	**1.89**	0.970
0.25	0.598	**0.58**	0.719	**0.91**	0.818	**1.24**	0.892	**1.57**	0.941	**1.90**	0.971
0.26	0.602	**0.59**	0.722	**0.92**	0.821	**1.25**	0.894	**1.58**	0.942	**1.91**	0.971
0.27	0.606	**0.60**	0.725	**0.93**	0.823	**1.26**	0.896	**1.59**	0.944	**1.92**	0.972
0.28	0.610	**0.61**	0.729	**0.94**	0.826	**1.27**	0.897	**1.60**	0.945	**1.93**	0.973
0.29	0.614	**0.62**	0.732	**0.95**	0.828	**1.28**	0.899	**1.61**	0.946	**1.94**	0.973
0.30	0.617	**0.63**	0.735	**0.96**	0.831	**1.29**	0.901	**1.62**	0.947	**1.95**	0.974
0.31	0.621	**0.64**	0.738	**0.97**	0.833	**1.30**	0.903	**1.63**	0.948	**1.96**	0.975
0.32	0.625	**0.65**	0.742	**0.98**	0.836	**1.31**	0.904	**1.64**	0.949	**1.97**	0.975

გაგრძელება

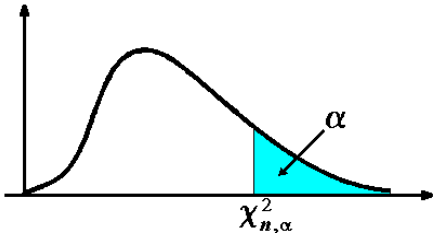
x	$\Phi(x)$	x	$\Phi(x)$	x	$\Phi(x)$	x	$\Phi(x)$	x	$\Phi(x)$
1.98	0.976	**2.26**	0.988	**2.54**	0.994	**2.82**	0.997	**3.10**	0.999
1.99	0.976	**2.27**	0.988	**2.55**	0.994	**2.83**	0.997	**3.11**	0.999
2.00	0.977	**2.28**	0.988	**2.56**	0.994	**2.84**	0.997	**3.12**	0.999
2.01	0.977	**2.29**	0.988	**2.57**	0.994	**2.85**	0.997	**3.13**	0.999
2.02	0.978	**2.30**	0.989	**2.58**	0.995	**2.86**	0.997	**3.14**	0.999
2.03	0.978	**2.31**	0.989	**2.59**	0.995	**2.87**	0.997	**3.15**	0.999
2.04	0.979	**2.32**	0.989	**2.60**	0.995	**2.88**	0.998	**3.16**	0.999
2.05	0.979	**2.33**	0.990	**2.61**	0.995	**2.89**	0.998	**3.17**	0.999
2.06	0.980	**2.34**	0.990	**2.62**	0.995	**2.90**	0.998	**3.18**	0.999
2.07	0.980	**2.35**	0.990	**2.63**	0.995	**2.91**	0.998	**3.19**	0.999
2.08	0.981	**2.36**	0.990	**2.64**	0.995	**2.92**	0.998	**3.20**	0.999
2.09	0.981	**2.37**	0.991	**2.65**	0.995	**2.93**	0.998	**3.21**	0.999
2.10	0.982	**2.38**	0.991	**2.66**	0.996	**2.94**	0.998	**3.22**	0.999
2.11	0.982	**2.39**	0.991	**2.67**	0.996	**2.95**	0.998	**3.23**	0.999
2.12	0.983	**2.40**	0.991	**2.68**	0.996	**2.96**	0.998	**3.24**	0.999
2.13	0.983	**2.41**	0.992	**2.69**	0.996	**2.97**	0.998	**3.25**	0.999
2.14	0.983	**2.42**	0.992	**2.70**	0.996	**2.98**	0.998	**3.26**	0.999
2.15	0.984	**2.43**	0.992	**2.71**	0.996	**2.99**	0.998	**3.27**	0.999
2.16	0.984	**2.44**	0.992	**2.72**	0.996	**3.00**	0.998	**3.28**	0.999
2.17	0.985	**2.45**	0.992	**2.73**	0.996	**3.01**	0.998	**3.29**	0.999
2.18	0.985	**2.46**	0.993	**2.74**	0.996	**3.02**	0.998	**3.30**	0.999
2.19	0.985	**2.47**	0.993	**2.75**	0.997	**3.03**	0.998	**3.31**	0.999
2.20	0.986	**2.48**	0.993	**2.76**	0.997	**3.04**	0.998	**3.32**	0.999
2.21	0.986	**2.49**	0.993	**2.77**	0.997	**3.05**	0.998	**3.33**	0.999
2.22	0.986	**2.50**	0.993	**2.78**	0.997	**3.06**	0.998	**3.34**	0.999
2.23	0.987	**2.51**	0.993	**2.79**	0.997	**3.07**	0.998	**3.35**	0.999
2.24	0.987	**2.52**	0.994	**2.80**	0.997	**3.08**	0.998	**3.36**	0.999
2.25	0.987	**2.53**	0.994	**2.81**	0.997	**3.09**	0.999	**3.37**	0.999

2. სტანდარტული ნორმალური განაწილების ზედა α-კრიტიკული წერტილების მნიშვნელობების ცხრილი



α	0.1	0.05	0.025	0.125	0.01	0.005	0.0025	0.001
z_α	1.28	1.64	1.96	2.24	2.33	2.57	2.81	3.08

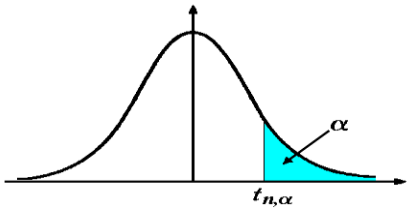
3. χ^2 -განაწილების ზედა α-კრიტიკული წერტილების მნიშვნელობების ცხრილი



	α							
n	**0.99**	**0.975**	**0.95**	**0.9**	**0.1**	**0.05**	**0.025**	**0.01**
1	0.0002	0.0010	0.0039	0.0158	2.7055	3.8415	5.0239	6.6349
2	0.0201	0.0506	0.1026	0.2107	4.6052	5.9915	7.3778	9.2104
3	0.1148	0.2158	0.3518	0.5844	6.2514	7.8147	9.3484	11.3449
4	0.2971	0.4844	0.7107	1.0636	7.7794	9.4877	11.1433	13.2767
5	0.5543	0.8312	1.1455	1.6103	9.2363	11.0705	12.8325	15.0863
6	0.8721	1.2373	1.6354	2.2041	10.6446	12.5916	14.4494	16.8119
7	1.2390	1.6899	2.1673	2.8331	12.0170	14.0671	16.0128	18.4753
8	1.6465	2.1797	2.7326	3.4895	13.3616	15.5073	17.5345	20.0902
9	2.0879	2.7004	3.3251	4.1682	14.6837	16.9190	19.0228	21.6660
10	2.5582	3.2470	3.9403	4.8652	15.9872	18.3070	20.4832	23.2093
11	3.0535	3.8157	4.5748	5.5778	17.2750	19.6752	21.9200	24.7250
12	3.5706	4.4038	5.2260	6.3038	18.5493	21.0261	23.3367	26.2170
13	4.1069	5.0087	5.8919	7.0415	19.8119	22.3620	24.7356	27.6882
14	4.6604	5.6287	6.5706	7.7895	21.0641	23.6848	26.1189	29.1412
15	5.2294	6.2621	7.2609	8.5468	22.3071	24.9958	27.4884	30.5780
16	5.8122	6.9077	7.9616	9.3122	23.5418	26.2962	28.8453	31.9999
17	6.4077	7.5642	8.6718	10.0852	24.7690	27.5871	30.1910	33.4087
18	7.0149	8.2307	9.3904	10.8649	25.9894	28.8693	31.5264	34.8052
19	7.6327	8.9065	10.1170	11.6509	27.2036	30.1435	32.8523	36.1908
20	8.2604	9.5908	10.850	12.442	28.412	31.410	34.169	37.566

			8	6	0	4	6	3
21	8.8972	10.2829	11.5913	13.2396	29.6151	32.6706	35.4789	38.9322
22	9.5425	10.9823	12.3380	14.0415	30.8133	33.9245	36.7807	40.2894
23	10.1957	11.6885	13.0905	14.8480	32.0069	35.1725	38.0756	41.6383
24	10.8563	12.4011	13.8484	15.6587	33.1962	36.4150	39.3641	42.9798
25	11.5240	13.1197	14.6114	16.4734	34.3816	37.6525	40.6465	44.3140
26	12.1982	13.8439	15.3792	17.2919	35.5632	38.8851	41.9231	45.6416
27	12.8785	14.5734	16.1514	18.1139	36.7412	40.1133	43.1945	46.9628
28	13.5647	15.3079	16.9279	18.9392	37.9159	41.3372	44.4608	48.2782
29	14.2564	16.0471	17.7084	19.7677	39.0875	42.5569	45.7223	49.5878
30	14.9535	16.7908	18.4927	20.5992	40.2560	43.7730	46.9792	50.8922

4. ***t*** - განაწილების ზედა α-კრიტიკული წერტილების მნიშვნელობების ცხრილი



	α						
n	**0.1**	**0.05**	**0.025**	**0.01**	**0.005**	**0.0025**	**0.001**
1	3.078	6.314	12.706	31.821	63.656	127.321	318.289
2	1.886	2.920	4.303	6.965	9.925	14.089	22.328
3	1.638	2.353	3.182	4.541	5.841	7.453	10.214
4	1.533	2.132	2.776	3.747	4.604	5.598	7.173
5	1.476	2.015	2.571	3.365	4.032	4.773	5.894
6	1.440	1.943	2.447	3.143	3.707	4.317	5.208
7	1.415	1.895	2.365	2.998	3.499	4.029	4.785
8	1.397	1.860	2.306	2.896	3.355	3.833	4.501
9	1.383	1.833	2.262	2.821	3.250	3.690	4.297
10	1.372	1.812	2.228	2.764	3.169	3.581	4.144
11	1.363	1.796	2.201	2.718	3.106	3.497	4.025
12	1.356	1.782	2.179	2.681	3.055	3.428	3.930
13	1.350	1.771	2.160	2.650	3.012	3.372	3.852
14	1.345	1.761	2.145	2.624	2.977	3.326	3.787
15	1.341	1.753	2.131	2.602	2.947	3.286	3.733

16	1.337	1.746	2.120	2.583	2.921	3.252	3.686
17	1.333	1.740	2.110	2.567	2.898	3.222	3.646
18	1.330	1.734	2.101	2.552	2.878	3.197	3.610
19	1.328	1.729	2.093	2.539	2.861	3.174	3.579
20	1.325	1.725	2.086	2.528	2.845	3.153	3.552
21	1.323	1.721	2.080	2.518	2.831	3.135	3.527
22	1.321	1.717	2.074	2.508	2.819	3.119	3.505
23	1.319	1.714	2.069	2.500	2.807	3.104	3.485
24	1.318	1.711	2.064	2.492	2.797	3.091	3.467
25	1.316	1.708	2.060	2.485	2.787	3.078	3.450
26	1.315	1.706	2.056	2.479	2.779	3.067	3.435
27	1.314	1.703	2.052	2.473	2.771	3.057	3.421
28	1.313	1.701	2.048	2.467	2.763	3.047	3.408
29	1.311	1.699	2.045	2.462	2.756	3.038	3.396
30	1.310	1.697	2.042	2.457	2.750	3.030	3.385

6. შემთხვევითი რიცხვების ცხრილები

6. ა. 400 შემთხვევითი ციფრი

86515	90795	66155	66434	56558	12332	94377	57802
69186	3393	42502	99224	88955	53758	91641	18867
41686	42163	85181	38967	33181	72664	53807	00607
86522	47171	88059	89342	67248	9082	12311	90316
72587	93000	89688	78416	27589	99528	14480	50961
52452	42499	33346	83935	79130	90410	45420	77757
76773	97526	27256	66447	25731	37525	16287	66181
04825	82134	80317	75120	45904	75601	70492	10274
87113	84778	45863	24520	19976	04925	7824	76044
84754	57616	38132	64294	15218	49286	89571	42903

6. ბ. 88 ნორმალური შემთხვევითი რიცხვი

0,2005	1,1922	-0,007	0,0348	1,0423	-1,8149	1,1803	0,0033
1,1609	-0,6690	-,5893	0,5816	1,8818	0,7390	-0,2736	1,0828
0,5864	-0,9245	0,0904	1,5068	-1,1147	0,2776	0,1012	-1,3566
0,1425	-0,2863	1,2809	0,4043	0,6379	-0,4428	-2,3006	-0,6446
0,9516	-1,7708	2,8854	0,4686	1,4664	1,6852	-0,9690	-0,0831
-0,5863	0,8574	-0,5557	0,8115	-0,2676	-1,2496	-1,2125	1,3846
1,1572	0,9990	-0,1032	0,5405	-6022	0,0093	0,2119	-1,4647
-0,4428	-0,5564	-0,5098	-1,1929	-0,0572	-0,5061	-0,1557	-1,2384
-0,3924	1,7981	0,6141	-1,3596	1,4943	-0,4406	-0,2033	-0,1316
0,8319	0,4270	-0,8888	0,4167	-0,8513	1,1054	1,2237	-0,7003
0,9780	-0,7679	0,8960	0,5154	-0,7165	0,8563	-1,1630	1,8800

ლიტერატურა

1. Averill M. Law and W. David Kelton, “Simulation, Modeling and Analysis”, OSBORNE, 2000;
   Аверилл М. Лоу, В. Давид Кельтон, “Имитационное Моделирование”, Питер, 2004;
2. Donald E. Knuth, “The Art of Computer Programming, vol. 2: Seminumerical Algorithms”, 3rd ed., Addison-Weasley, Reading, Massachussets, 1998;
   Д. Кнут, Искуство программирования для ЭВМ, т. 2, Получисленные алгоритмы, Изд-во Мир, 1977;
3. Donald E. Knuth, “The Art of Computer Programming, vol. 3: Sorting and Searching”, 2nd ed., Addison-Weasley, Reading, Massachussets, 1998;
4. У. Гренандер, В. Фрайбертер, “Краткий Курс Вычислительной Вероятностей и Математической Статистики”, М.: “Наука”, 1978;
5. В.Е. Гмурман, “Руководство к Решению Задач по Теории Вероятностей и Математической Статистике”, М.: “Высшая Школа”, 1998;
6. И.М. Соболь, “Метод Монте-Карло”, М.: “Наука”, 1985;
7. В. Феллер, “Введение в Теорию Вероятностей и Ее Приложения”, т. 1, М.: “Мир”, 1984.

შ ი ნ ა ა რ ს ი

სისტემების მოდელირება და სიმულაცია