F/A-18 High Alpha fighter testing high angle-of-attack aerodynamics in 80x120ft W.T.

2 MATLAB, შესავალი

შესავალი

მოდელირება არის აბსტრაქტული, კონცეპტუალური გრაფიკული ან მათემატუკური მოდელის შექმნა. მეცნიერება გვთავაზობს უამრავ მეთოდს, ტექნიკასა და თეორიას მოდელირებისათვის.

მოდელირება ძირითადი, გაუყოფელი ნაწილია თანამედროვე სამეცნიერო საქმიანობაში. თიტოეულ სამეცნიერო სფეროს თავისი სპეციფიკური მეთოდები გააჩნია მოდელირებისათვის.

მოდელი ნიშნავს ობიექტის, მოვლენისა თუ ფიზიკური პროცესის ემპირიულ წარმოდგენას ლოგიკური და ობიექტური გზით. მოდელი არის რეალობის გამარტივებული ასახვა. მაგრამ მიუხედავად მათი სიყალბისა, ზალზე მნიშვნელოვან ინფორმაციას იძლევა. მოდელის აგება სიმულირება ფუნდამენტურ როლს თამაშობს თანამედროვე მეცნიერებაში.

საზოგადოდ მოდელირებას მიმართავენ როცა შეუძლებელია, ან არაპრაქტიკულია ეხპერიმენტული პირობების შექმნა, სადაც მეცნიერებმა უნდა უშუალოდ შეაფასონ ეხპერომენტის შედეგი. კონტროლირებად პირობებში ჩატარებული ექსპერიმენტის შედეგები რა თქმა უნდა უფრო ზუსტია, ვიდრე მოდელირების. იმიტომ რომ მოდელირებისას ხშირ შემთხვევაში უამრავ დაშვებას მივმართავთ, რაც თავიდანაა აცილებული ეხპერიმენტის ჩატარებისას.

სიმულაცია არის მოდელის განხორციელება. მდგარდი მდგომარეობის სიმულაცია გვაწვდის ინფორმაციას დროის გარკვეულ მომენტში, ხოლო დინამიური სიმულაცია იძლევა

ინფორმაციას დროის გარკვეულ შუალედში. სიმულაციას მოდელი მოქმედებაში მოყავს და გვიჩვენებს როგორ იქცევა კონკრეტული ობიექტი, ან რა მოსდის გარკვეულ პროცესს დროში. სიმულაცია გამოიყენება ტესტირებისა და ანალიზისათვის როცა რეალური სისტემა ან კონცეპცია შესაძლებელია წარმოვადგინოთ მოდელის სახით.

მოდელი ძირითადად იყენებს მოვლენის მხოლოდ ზოგიერთ ასპექტს და ორი სხვადასხვა მოდელი ერთიდაიგივე მოვლენისა, შესაძლოას სრულიად განსხვავდებოდეს ერთმანეთისაგან. მოდელის აგებისას კარგად უნდა გვესმოდეს მოდელირების ძირითადი მიზანი და ის დაშვებანი, რასაც ვიყენებთ მოდელირებისას

კომპიუტერი ძირითადი საშუალებაა საინჟინრო პრობლემების ამოხსნის საქმეში. ამ თავში მოკლედ მიმოვიხილავთ გამოთვლით სისტემებს, მათ შორის კომპიუტერის ტექნიკურ(ინსტრუმენტალ) აღჭურვილობას და პროგრამულ უზრუნველყოფას. შემდეგ შევუდგებით MATLAB-ს, რომელიც გახლავთ წამყვანი პროგრამული პაკეტი ინტერაქტიული რიცხვითი გამოთვლების, მონაცემთა ანალიზის და მათი გრაფიკული ინტერპრეტაციისათვის. მოკლე შესავლის შემდეგ განვიხილავთ MATLAB-ში მონაცემთა წარმოდგენის საშუალებებს, შემდეგ არსებული მონაცემების ბაზაზე ავაგებთ გრაფიკს.

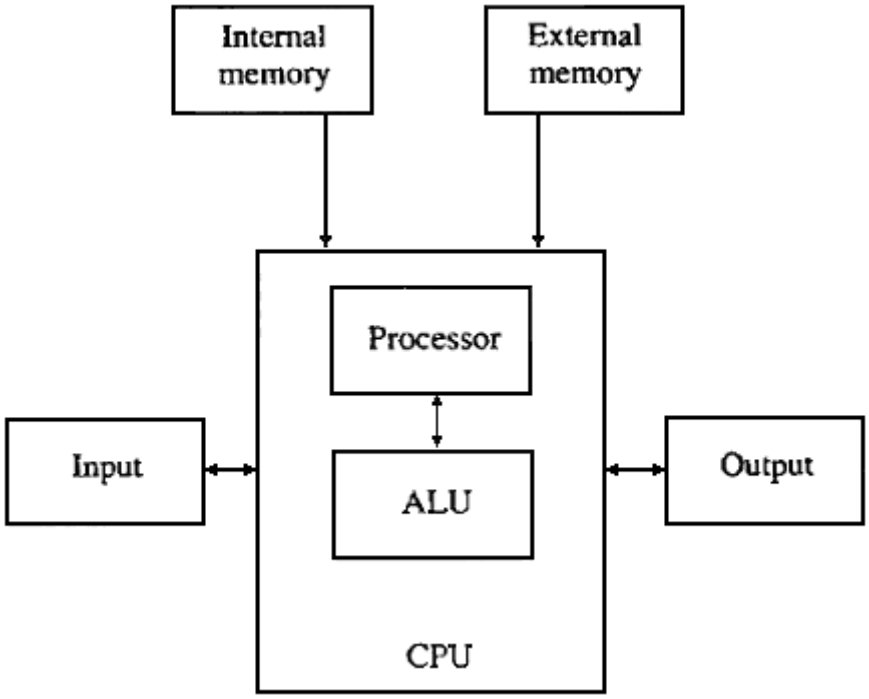
2.1 გამოთვლითი სისტემები

ვიდრე შევუდგებით MATLAB-ის შესწავლას, მოკლედ მიმოვიხილოთ კომპიუტერული სისტემები. კომპიუტერი არის მოწყობილობა, რომელიც შეიქმნა იმისათვის, რომ აწარმოოს ოპერაციები, რომელნიც განსაზღვრულია ინსტრუქციების მწკრივით, ეგრეთ წოდებული პროგრამით. კომპიუტერის აღჭურვილობას (HARDWARE) წარმოადგენს ისეთი მოწყობილობანი, როგორიცაა კლავიატურა, მაუსი, მყარი დისკი, პრინტერი და სხვა. კომპიუტერის პროგრამული უზრუნველყოფა (SOFTWARE) წარმოადგენს პროგრამას, რომელიც მოიცავს იმ ბრძანებებს, რაც კომპიუტერმა უნდა შეასრულოს.

კომპიუტერის აღჭურვილობა

ყველა კპმპიუტერს აქვს ზოგადად ერთნაირი შიდა ორგანიზაცია რომელიც ნაჩვენებია ნახ. 2.1 პროცესორი არის კომპიუტერის ნაწილი, რომელიც ყველა სხვა დანარჩენს აკონტროლებს. ის ღებულობს შესავალ მნიშვნელობებს (მაგალითად კლავიატურიდან) და ათავსებს მათ მეხსიერებაში. ის აგრეთვე ინტერპრეტაციას უკეთებს პროგრამაში მითითებულ ინსტრუქციებს. თუ გვინდა ერთმანეთს მივუმატოთ ორი სიდიდე, კომპიუტერმა უნდა მოძებნოს ეს სიდიდეები მეხსიერებაში და გააგზავნოს ის არითმეტიკულ ლოგიკურ მოწყობილობაში (ALU). ALU შეასრულებს შეკრებას და პროცესორი მიღებულ შედეგს მოათავსებს მეხსიერებაში. პროცესორი და ALU იყენებენ მეხსიერების მცირე ნაწილს-შიდა მეხსიერებას. მონაცემთა უმრავლესობა განთავსებულია გარე მეხსიერებაში-მყარი დისკი, მაგნიტური დისკი, რომლებიც დაკავშირებულია პროცესორთან. პროცესორი და ALU ერთად წარმოადგენს ცენტრალურ პროცესორს – CPU. მიკროპროცესორი არის CPU, რომელიც მოთავსებულია მცირე ზომის მიკროსქემაში (integrated circuit chip), რომელიც ასევე შეიცავს ათასობით სხვა კომპონენტს და მისი ზომა დაახლოებით თქვენი ფრჩხილისოდენაა.

გამოთვლილი სიდიდეები შესაძლებელია დავინახოთ მონიტორზე, ან დავბეჭდოთ პრინტერზე. (ლაზერული, მატრიცული, ჭავლური). მონაცემები შეგვიძლია აგრეთვე ჩავწეროთ მაგნიტურ დისკზე. დაბეჭდილ ინფორმაციას უწოდებენ მყარ ასლს (HARDCOPY), ხოლო მაგნიტურ ასლს – ელექტრონულ ასლს.



ნახ. 2.1 კომპიუტერის შიდა ორგანიზაცია

არსებობს სხვადასხვა ზომის და ფორმის კომპიუტერი. პრაქტიკაში უფრო ხშირად გვხვდება პერსონალური კომპიუტერი. უფრო მძლავრი მინიკომპიუტერი და უნივერსალური კომპიუტერი ძირითადად გამოიყენება ლაბორატორიებში, ქსელური კომპიუტერი (WORKSTATION), სუპერკომპიუტერი - უსწრაფესი კომპიუტერი, რომელიც გამოიყენება ურთულესი ამოცანების ამოსახსნელად, რომელთაც სხვა ტიპის კომპიუტერი ვერ უმკლავდება.

კომპიუტერი უნდა შეირჩეს იმ ამიცანის შესაბამისად, რომლის ამოხსნასაც ვისახავთ მიზნად. მაგალითად თუ კომპიუტერი საშინაო უსაფრთხოების სისტემის ნაწილია, მიკროპროცესორიც საკმარისია, მაგრამ თუ მიზნად ვისახავთ რაიმე პროცესის მოდელირებას, უნივერსალური კომპიუტერია საჭირო. კომპიუტერული ქსელი სასარგებლოა იმით, რომ შესაძლებელია მათ შორის მონაცემთა გაცვლა და მათი რესურსების გაერთიანება.

პროგრამული უზრუნველყოფა

პროგრამული უზრუნველყოფა შეიცავს ინსტრუქციებს, ბრძანებებს, რომელიც კომპიუტერმა უნდა შეასრულოს. არსებობს პროგრამული უზრუნველყოფის რამდენიმე მნიშვნელოვანი კატეგორია. განვიხილოთ ისინი:

ოპერაციული სისტემა. ოპერაციული სისტემა პროგრამული უზრუნველყოფის ის ფორმაა, რომელიც საშუალებას გვაძლევს ვიურთიერთოთ კომპიუტერთან. ეს არის ინტერფეისი, მოხერხებული გარემო, სადაც შეგვიძლია შევარჩიოთ და გავუშვათ სხვადასხვა პროგრამა. ოპერაციულ სისტემას აქვს პროგრამათა ჯგუფი, ე.წ. utilities, რომლებიც საშუალებას გვაძლევს შევასრულოთ ფაილების ორგანიზება, კოპირება, დაბეჭდვა. ეს პროცესი საერთოა ყველა სახის ოპერაციული სისტემისათვის, თუმცა მათი განხორციელების ფორმა განსხვადება.

პროგრამული საშუალებები. პროგრამული საშუალებები არის პროგრამები, რომელთა საშუალებით ხორციელდება ზოგადი ოპერაციები. მაგალითად Microsoft Word და Word Perfect გვეხმარება შევქმნათ ტექსტი და მოვახდინოთ მისი ფორმატირება, წავშალოთ სიტყვები და წინადადებები, შევცვალოთ შრიფტი და შევამოწმოთ ტექსტის მართლწერა.

ცხრილური პროგრამები საშუალებას გვაძლევს წარმოვადგინოთ ჩვენს ხელთ არსებული მონაცემები ცხრილის სახით. ეს პროგრამები თავიდან შექმნილი იყო ფინანსური და საბუღალტრო მონაცემებისათვის, მაგრამ მათ დიდი გამოყენება ჰპოვეს მეცნიერებაშიც. ცხრილურ პროგრამებს აქვთ აგრეთვე გრაფიკული საშუალებებიც, რაც მათ უფრო მოხერხებულს ხდის.

ერთერთი პოპულარული პროგრამული საშუალებაა მონაცემთა ბაზის მართვის პროგრამა, რომელიც საშუალებას იძლევა ვიმუშაოთ მონაცემთა ვრცელ სისტემასთან. მარტივად ამოვიღოთ მონაცემთა ჩვენთვის სასურველი ნაწილი. ამ პროგრამას ფართოდ იყენებენ ბანკებში, სასტუმროებსა თუ მომსახურების სხვა სისტემაში. მას ფართოდ იყენებენ მეცნიერებაშიც, მაგალითად, მეტეოროლოგიური მონაცემები მაგალითია სამეცნიერო მონაცემევისა, რომელიც მოითხოვს მონაცემთა ვრცელი ბაზის შექმნას და შემდგომში მის მეცნიერულ ანალიზს.

გრაფიკული პაკეტი საშუალებას იძლევა ავაგოთ მრავალფეროვანი გრაფიკები, ორგანზომილებიანი, სამგანზომილებიანი, ბარ და კონტურული გრაფიკები. KAD (Komputer-aidid design) პაკეტებს, როგორიცაა AutoCAD, AutoSketch და MathCAD აქვთ აგრეთვე მრავალფეროვანი გრაფიკული შესაძლებლობანი.

არესებობს აგრეთვე მძლავრი კომპიუტერული გამოთვლითი საშუალებები, როგორიცაა MATLAB, Mathematika. გარდა იმისა, რომ მათ გაჩნიათ მათემატიკური გამოთვლების მძლავრი საშუალებები, ფართო გრაფიკულ შესაძლებლობებსაც ფლობენ. ეს კომბინაცია მძლავრი მათემატიკური აპარატისა და ვიზუალიზაციისა, განაპირობებს ამ პროგრამული პაკეტების ფართო გამოყენებას საინჟინრო ამოცანების გადაწყვეტაში.

კომპიუტერული ენა. ეს არის ენა, რომელიც "ესმის" კომპიუტერს. დღეისათვის კპმიუტერის მუშაობა ემყარება ორსაფეხურიან (two-state) ტექნოლოგიას. (მოწყობილობა ორი მდგომარეობით როგორც ღია ან შეკრული წრედი, ან ჩართული და გამორთული, უარყოფითი ან დადებითი მუხტი), მანქანური ენა იწერება ორი სიმბოლოს საშუალებით 0 და 1. მანქანის ენა არის ორობოთი და ინსტრუქციები იწერება 0 და 1 შედგენილი მწკრივის სახით - ორობითი სტრიქონი. მანქნურ ენას უწოდებენ აგრეთვე დაბალი დონის ენას და იგი დამოკიდებულია კომპიუტერის სტრუქტურაზე, მაგ. SAN მანქანების ენა განსხვავდება VAX კომპიუტერების მანქანური ენისაგან.

მაღალი დონის ენას უწოდებენ კომპიუტერულ ენას, რომელსაც აქვს ჩვეულებრივი სალაპარაკო ინგლისური ენის მსგავსი ბრძანებები და ინსტრუქციები. ესენია C, Fortran, Pascal, Basic. პროგრამის დაწერა მაღალი დონის ენაზე რა თქმა უნდა გაცილებით მარტივია, ვიდრე დაბალი დონის ენაზე. მაღალი დონის ენაზე დაწერილი პროგრამა უნდა შეიცავდეს დაწვრილებით ინსტრუქციებს იმის შესახებ თუ რა უნდა გააკეთოს კომპიუტერმა. გარდა ამისა მაღალი დონის ენას გაჩნია უამრავი ბრძანება და სინტაქსი.

თუ შესაძლებელია პრობლემის ამოხსნა პროგრამული საშუალებებით, უმჯობესია გამოვიყენოთ ეს შესაძლებლობა, რადგან ეს მისი გადაჭრის უსწრაფესი გზაა. მაგრამ თუ სისტემური საშუალებებით არ ხერხდება დასახული ამოცანის გადაჭრა, კომპიუტერული

ენები საშუალებას გვაძლევს დავწეროთ ახალი პროგრამა ჩვენთვის სასურველი საკითხის გადასაწყვეტად.

2.2 ზოგადი ინფორმაცია MATLAB-ის შესახებ

MATLAB-ი შეიქმნა როგორც "მატრიცული ლაბორატორია". დღეისათვის იგი წარმოადგენს იტერაქტიულ სისტემურ და პროგრამულ ენას ფართო სამეცნიერო და ტექნიკური გამოთვლებისათვის. მისი ძირითადი ელემენტია მატრიცა. MATLAB-ში პროგრამის დაწერა გაცილებით მარტივია, ვიდრე რომელიმე მაღალი დონის ენაზე. ამ თავში განვიხილავთ რამდენიმე ძირითად ინფორმაციას სამუშაო სივრცის (workspace) შესახებ.

სტუდენტური ვერსია

სტუდენტური ვერსია პროფესიულის იდენტურია გარდა რამდენიმე მახასიათებლისა.

- ყოველი ვექტორი ან მატრიცა შეიცავს არაუმეტეს 1024 ელემენტისა
- მეტაფაილი და გრაფიკული პოსტპროცესორი გრაფიკების დასაბეჭდად შეუძლებელია
- მათემატიკური თანაპროცესორი არ არის საჭირო, მაგრამ შესაძლოა გამოყენებული იქნას, თუ შესაძლებელია
- სიგნალის და სისტემური ტულბოქსი

სამუშაო სივრცე (**workspace**). იმისათვის რომ შევუდგეთ მუშაობას MATLAB–ში, შეარჩიეთ იგი თქვენს ოპერაციულ სისტემაში ან აკრიფეთ MATLAB კლავიატურაზე. დაინახავთ ნიშანს (>>), რაც იმის მანიშნებელია, რომ MATLAB –ი ელოდება თქვენს ბრძანებას. თუ ეს თქვენი პირველი ნაბიჯია MATLAB–ში, ისარგებლეთ ბრძანებით **demo** იმისათვის რომ ნახოთ სადემონსტრაციო პროგრამების სია, რომელიც შეგიძლიათ გაუშვათ და ნახოთ MATLAB-ის შესაძლებლობები. იმისათვის, რომ დაასრულოთ მუშაობა MATLAB-ში ისარგებლეთ ბრძანებებით: **quit** ან **exit**. მაგრამ თუ გსურთ შეინახოთ თქვენს მიერ შექმნილი ცვლადები მუშაობის დამთავრებამდე, ისარგებლეთ ბრძანებით **save**. ასე შეინახავთ ცვლადებს ფაილში, რომლის სახელი იქნება matlab.mat. როცა ხელახლა გახსნით MATLAB-ს, შეგიძლიათ აღადგინოთ ცვლადები ბრძანებით **load**. ბრძანება **computer** გვაძლევს ინფორმაციას კომპიუტერის ტიპის შესახებ.

MATLAB-ს აქვს ბრძანებების ფანჯარა, რომელიც საშუალებას გვაძლევს შევიყვანოთ ბრძანებები და მივიღოთ შედეგები იქვე, ეკრანზე ან დავბეჭდოთ და გრაფიკების ფანჯარა. ორივე ფანჯარა ცარიელია, როცა იწყებთ მუშაობას. თუ გსურთ გაასუფთაოთ ბრძანებების ფანჯარა მუშაობის პროცესში, უნდა გამოიყენოთ ბრძანება **clc**. გრაფიკების ფანჯრის გასასუფთავებლად გამოიყენება ბრძანება ცლფ. ასევე შესაძლებელია გავასუფთაოთ სამუშაო სივრცე – **clear**. ამით ყველა ცვლადი, რომელიც შეიქმნა მოცემულ სამუშაო სივრცეში, წაიშლება.

მნიშვნელოვანია ვიცოდეთ როგორ შევწყვიტოთ უკვე მიცემული ბრძანება. მაგალითად როცა რომელიმე ბრძანების გამო კომპიუტერი გაუთავებლად იმეორებს რაიმე ციკლს, ჩაიციკლება. პროცესის შესაწყვეტად ისარგებლეთ ბრძანებით **ctrl+c**.

იმისათვის, რომ გავიგოთ ჩვენს მიერ შექმნილი ცვლადების და მატრიცების სტატუსი MATLAB-ს აქვს რამდენიმე ბრძანება. ბრძანება **who** გვაძლევს ცვლადების სიას, რომელიც შევქმენით, ხოლო **whos** დამატებით ინფორმაციასაც იძლევა მათი ზომის, მანქანური მოცულობის და კლასის შესახებ.

ეს ბრძანებები ცვლადებში მოიხსენიებენ ცვლადს სახელით ანს, რომელიც შესაძლოს თქვენ არ შეგიქმნიათ. ეს სახელი გამოიყენება ცვლადისათვის, რომელიც გამოვითვალეთ, მაგრამ სახელი არ მივანიჭეთ. ბრძანება **size** საჭიროა თუ გვსურს გავიგო რომელიმე ცვლადის ზომა. მაგ size(A).

MATLAB-ის ბრძანებები ჩვეულებრივ იკრიფება ახალ სტრიქონში, მაგრამ შეგვიძლია რამდენიმე ბრძანება მივცეთ ერთად, თუ მათ წერტილ-მძიმით გამოვყოფთ. ტექსტი, რომელიც მოყვება ნიშანს **%** MATLAB-ის მიერ იგნორირებულია და გამოიყენება კომენტარისათვის. ჩვენ გამოვიყენებთ ამ ნიშანს ჩვენს მაგალითებში კომენტარებისათვის, მაგალითად ცვლადების გამნმარტებისთვის.

MATLAB-ის მნიშვნელოვანი რესურსია **help** ბრძანება. მას მოჰყვება სია იმ საკითხებისა, რომელთათვისაც შეგვიძლია ვისარგებლოთ ამ ბრძანებით.

M – ფაილები

გარდა იმისა, რომ ბრძანებები შეგვიძლია მივცეთ ბრძანებების ფანჯრიდან, შეგვიძლია წინასწარ შევქმნათ ბრძანებათა მწკრივი ტექსტური ფაილის სახით, რომლის სახელს უნდა ჰქონდეს გაფართოება .m, მაგ. labtest.m ასეთ ფაილებს ეწოდება **M ფაილები.** ფაილის სახით შენახული ბრძანებათა ასეთი მწკრივი შეგვიძლია შემდგომშიც გამოვიყენოთ. ამ ფაილებს აგრეთვე უწოდებენ ხელნაწერ ფაილებს, რადგან წარმოადგენს ტექსტურ ASCII ფაილს და შეგვიძლია შევქმნათ ტექსტური რედაქტორით ან წორდ პროცესორით. თუ გვინდა ბრძანებების ფანჯარაში გამოვიყვანოთ .m ფაილის ტექსტი უნდა ვისარგებლოთ ბრძანებით **echo**

M ფაილები ასევე გამოიყენება MATLAB-ის ახალი ფუნქციის შესაქმნელად.

ბრძანება **what** გვაძლევს იმ ფაილების სიას, რომლების განთავსებულია მიმდინარე დირექტირიაში. ბრძანება **type** გვაძლევს მითითებული ფაილის შინაარსს.

2.3 მატრიცა, ვექტორი, სკალარი

საინჟინრო ამოცანის ამოხსნისას მნიშვნელოვანია მოვახდინოთ პრობლემასთან დაკავშირებული მონაცემების ვიზუალიზაცია. ზოგჯერ ეს მონაცემი ერთი რიცხვია, მაგ. წრის რადიუსი. სხვა შემთხვევაში ეს შესაძლოა იყოს წერტილის კოორდინატები (x,y) სიბრტყეზე, ანდა შეიძლება გვქონდეს x-y-z კოორდინატების 4 წყება, რომელიც წარმოადგენს სივრცეში პირამიდის ოთხ წვეროს. ეს მონაცემები შეგვიძლია წარმოვადგინოთ მონაცემთა სპეციალური სტრუქტურის – მატრიცის სახით. მატრიცა ეს არის მონაცემთა მწკრივი დალაგებული სტრიქონების და სვეტების სახით. ერთი წერტილი შეგვიძლია განვიხილოთ როგორც მატრიცა, რომელიც შედგება ერთი სტრიქონისა და ერთი სვეტისაგან, x, y წყვილი – მატრიცა ერთი სტრიქონით და ორი სვეტით, ხოლო x-y-z კოორდინატების 4 წყება – მატრიცა ოთხი სტრიქონისა და სამი სვეტისაგან. მაგალითად:

```
A = [3.5]
B = [1.5 3.2]
```

$$C = \begin{matrix} -1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 2 \end{matrix}$$

თუ მატრიცა შეიცავს ერთ სტრიქონსა და ერთ სვეტს, იგი სკალარია, თუ იგი შეიცავს 1 სტრიქონს და რამდენიმე სვეტს – სტრიქონი ვექტორია, ხოლო თუ შეიცავს რამდენიმე სტრიქონსა და 1 სვეტს – იგი სვეტი ვექტორია.

მატრიცის ელემენტებს მიუთითებენ ორი ინდექსით – სტრიქონის და სვეტის ნომერი. მაგ. C მატრიცის მეოთხე სტრიქონის და მესამე სვეტის ელემენტია $C_{4,3}=0$.

მატრიცის ზომა განისაზღვრება სტრიქონების და სვეტების რაოდენობით. მაგ. C მატრიცის ზომაა 4×3. თუ მატრიცის სვეტების და სტრიქონების რაოდენობა ერთიდაიგივეა, მას კვადრატულს უწოდებენ.

სავარჯიშო

უპასუხეთ კითხვებს შემდეგი მატრიცის მაგალითზე:

$$C = \begin{bmatrix} 0.6 & 1.5 & 2.3 & -0.5 \\ 8.2 & 0.5 & -0.1 & -0.2 \\ 5.7 & 8.2 & 9.0 & 1.5 \\ 0.5 & 0.5 & 2.4 & 0.5 \\ 1.2 & -2.3 & -4.5 & 0.5 \end{bmatrix}$$

1. რა ზომისაა მატრიცა?
2. არის თუ არა იგი კვადრატული?
3. დაწერეთ აღნიშვნა მატრიცის ელემენტებისა, რომელთა მნიშვნელობა 0.5 ტოლია
4. დაწერეთ აღნიშვნა მატრიცის ელემენტებისა, რომელთა მნიშვნელობა უარყოფითია

მატრიცის ინიციალიზაცია

როგორ განვსაზღვროთ მატრიცა MATLAB-ში? განვიხილავთ ოთხ სხვადასხვა მეთოდს. პირველი მეთოდი ცხადად განვსაზღვროთ მატრიცის ელემენტები, მეორე მეთოდი – წავიკითხოთ მატრიცა ფაილიდან, მესამე – გამოვიყენოთ (ორწერტილიანი) კოლონოპერატორი (:), მეოთხე- შევიყვანოთ მონაცემები კლავიატურიდან.

პირველი მეთოდი – ცხადად განვსაზღვროთ მატრიცა. ჩავწეროთ კვადრატულ ფრჩხილებში მატრიცის ელემენტები:

```
A= [3.5];
B= [1.5, 3.1];
```

```
C= [-1, 0, 0; 1, 1, 0; 1,- 1, 0; 0, 0, 2];
```

ეს ბრძანებები მაგალითია, თუ როგორ შევქმნათ სხვადასხვა ზომის მატრიცა. მატრიცის სახელი უნდა იწყებოდეს ასოითი გამოსახულებით და შედგებოდეს არაუმეტეს 19 ნიშნისა (დასაშვებია ციფრები, ასოები და ქვედა ტირე) და უნდა ეწეროს უდრისის ნიშნის მარცხნივ. ტოლობის ნიშნის მარჯვენა მხარე წარმოადგენს კვადრატულ ფრჩხილებში ჩასმულ მატრიცის ელემენტების მნიშვნელობებს, წერტილმძიმით სტრიქონები გამოიყოფა ერთმანეთისაგან, ხოლო სტრიქონებში მონაცემები შეიძლება გამოვყოთ მძიმით ან მის გარეშე. გამოსახულება შეიძლება შეიცავდეს + ან – ნიშანს, ათწილადის ნიშანს “.”და არა “,”. როდესაც განვსაზღვრავთ მატრიცას, თუ ბოლოში არ დავუსვამთ წერტილ-მძიმეს, MATLB ბრძანებების ფანჯარაში გამოგვიყვანს მის გამოსახულებას. სცადეთ შეიყვანოთ ზემოთაღნითნული მონაცემები ბოლოში წერტილ-მძიმის დაურთველად.

მატრიცა შეგვიძლია განვსაზღვროთ ისე, რომ სრტიქონები წერტი-მძიმით კი არ გამოვყოთ ერთმანეთისგან, არამედ გადავიტანოთ შემდეგ ხაზზე “ ნტერ” კლავიშის საშუალებით.

```
C = [-1 0 0
     1 1 0
     1 –1 0
     0 0 2] ;
```

თუ მატრიცის სტრიქონში ბევრი ელემენტებია, შეგვიძლია გარკვეული რაოდენობით ელემენტების შემდეგ დავწეროთ სამი წერტილი და სტრიქონი გავაგრძელოთ შემდეგ ხაზზე. მაგალითად:

```
F = [ 1 0 5 7 3 9 6 7 8 0 4];
```

იგივეა რაც

```
F = [ 1 0 5 7 3 . . .
      9 6 7 8 0 4];
```

MATLAB საშუალებას იძლევა განვსაზღვროთ ახალი მატრიცა უკვე შექმნილის ბაზაზე. მაგალითად განვიხილოთ შემდეგი ბრძანებები:

```
B = [1.5, 3.1];
S = [3.0 B];
```

ეს იგივეა, რაც:

```
S = [3.0 1.5 3.1];
```

ასევე შეგვიძლია შევცვალოთ მატრიცის ელემენტების მნიშვნელობები, ან დავუმატოთ ახალი ელემენტები:

```
S(2) = -1.0;
```

ეს ბრძანება შეცვლის შ მატრიცის მეორე ელემენტს 1.5 მიანიჭებს რა მას ახალ მნიშვნელობას –1.0.

ასევე შესაძლებელია მატრიცას დავუმატოთ ახალი ელემენტები:

```
S(4) = 5.5;
```

ამის შედეგად სამელემენტიანი მატრიცა ოთხელემენტიანად იქცევა. მაგრამ თუ მივცემთ ბრძანებას:

```
S(8) = 9.5;
```

მატრიცა გახდება 8 ელემენტიანი, ხოლო მისი მეხუთე, მეექვსე და მეშვიდე ელემენტები მიიღებენ 0-ის ტოლ მნიშვნელობებს.

სავარჯიშო

განსაღვრეთ მტრიცის ზომა და შეამოწმეთ იგი MATLAB-ის საშუალებით:

1. A = [1, 0, 0, 0, 0, 1];
2. B = [2; 4; 6; 10];
3. C = [5 3 6; 6 2 –3];
4. D = [3 4

   05 7
   9 10];
5. E = [3 5 10 0; 0 0 . . .
   0 3; 3 9 9 8];
6. T = [4 24 9];
   Q = [T 0 T];
7. X = [3 6];
   Y = [D; X];
8. R = [C; X; 5];
9. V = [C(2,1); B];
10. A(2,1) = -3;

მატრიცა შეგვიძლია შევქმნათ აგრეთვე იმ ინფორმაციის საშუალებით, რომელიც ჩაწერილია ტექსტურ ფაილში – MAT, რომელიც შეიცავს ინფორმაციას ორობით ფორმატში ან ASCII ფაილში.

MAT ფაილი იქმნება MATLAB-ის მიერ **save** ბრძანების საშუალებით. მაგალითად ბრძანება

save data1 x y;

შეინახავს x და y მონაცემებს ფაილში data1.mat .mat გაფართოებას პროგრამა ავტომატურად დაურთავს ფაილის სახელს.

ამ მონაცემთა სამუშაო სივრცეში გამოსაძახებლად ვსარგებლობთ ბრძანებით:

load data1;

ASCII ფაილი, რომელიც გვინდა MATLAB –ის პროგრამაში გამოვიყენოთ, უნდა შეიცავდეს მხოლოდ რიცხვით მნიშვნელობებს, ამასთანავე, ყოველი სტრიქონი ელემენტთა ერთნაირ

რაოდენობას უნდა შეიცავდეს. ეს ფაილი შეიძლება შევქმნათ ნებისმიერი ტექსტური რედაქტორის საშუალებით და აგრეთვე MATLAB-ის მეშვეობით თუ გამოვიყენებთ ბრძანებას:

save data1.dat Z /ascii;

Z მატრიცის ყოველი სტრიქონი ჩაიწერება ფაილის ცალკე სტრიქონის სახით. mat გაფართოება არ დაერთვის ascii ფორმატის ფაილს, მაგრამ უმჯობესია ფაილს მივანიჭოთ dat გაფართოება რათა ვიზუალურად განვასხვაონ MAT და M ფაილებისაგან.

დავუშვათ ASCII ფაილი data1.dat შეიცავს დროის და შესაბამისი მანძილის მნიშვნელობათა მწკრივს. ამ ფაილის პირველი სტრიქონები დაახლოებით ასე უნდა გამოიყურებოდეს:

0.00	0.00
0.01	0.1255
0.02	0.2507

ბრძანება:

load data1.dat;

მიმართავს მითითებულ ფაილს და შექმნის ორსვეტიან მატრიცას data1 სამუშაო სივრცეში.

ორწერტილოვანი ოპერატორის საშუალებით შეგვიძლია მატრიცისაგან შევქმნათ ვექტორი. მაგალითად თუ გვსურს ავაგოთ გრაფიკი x-y მონაცემთა მიხედვით, მოხერხებულია თუ გვექნება x და y მონაცემები ვექტორის სახით. როცა ორი წერტილი გვაქვს მატრიცის ელემენტთა აღნიშვნაში, ის მიუთითებს ყვალა სტრიქონს ან ყველა სვეტს. მაგალითად თუ ავიღებთ ჩვენს მიერ განხილულ მონაცემებს ფაილიდან data1.dat შემდეგი ბრძანებები ააგებს ორ სვეტ ვექტორს x და y :

x = data1(: , 1);
y = data1(: , 2);

ეს ოპერატორი ასევე გამოიყენება ახალი მატრიცის საწარმოებლად. თუ იგი დგას ორ მთელ რიცხვს შორის, შეიქმნება ვექტორი, რომლის ელემენტები იქნება ყველა მთელი რიცხვი, რომელიც ამ ორ რიცხვს შორის მდემარეობს, მათი ჩათვლით. მაგალითად:

```
H = 1:8
```

იძლევა ვექტორს

```
H = [1 2 3 4 5 6 7 8];
```

თუ ორწერტილოვანი ოპერატორი 3 რიცხვს გამოყოფს ერთმანეთისაგან, მაშინ შეიქმნება ვექტორი, რომლის ელემენტები იქნება რიცხვები პირველსა და მესამე რიცხვს შორის, ნაზრდით მეორე რიცხვი:

```
Time = 0.0:0.5:5.0;
```

გვაძლევს

```
Time = [0.0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5];
```

ნაზრდი შეიძლება იყოს უარყოფითი.

ორწერტილოვანი ოპერატორი გამოიყენება აგრეთვე იმ შემთხვევაში, თუ გვინდა ამოვიღოთ მატრიცა უკვე შექმნილი მატრიციდან. დავუშვათ გვაქვს მატრიცა :

$$C = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

თუ მივცემთ ბრძანებებს:

```
C_PARTIAL = C( : , 2 : 3);

C_PARTIAL = C(3 : 4, 1 : 2);
```

მივიღებთ ორ ახალ მატრიცას:

$$C\_PARTIAL\_1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 2 \end{bmatrix} \qquad C\_PARTIAL\_2 = \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix}$$

თუ ორწერტილოვანი ოპერატორი გამოყენებულია მატრიცის არასწორი ინდექსისათვის, MATLAB მიგვითითებს შეცდომაზე.

MATLAB –ში ზოგჯერ გვჭირდება ცარიელი მატრიცა, იგი განსხვავდება მატრიცისაგან, რომლის ყველა ელემენტი ნულის ტოლია.

```
a = [];
b = 4 : -1 : 5;
```

ორივე ბრძანება ქმნის ცარიელ მატრიცას.

გამოსახულება C (:) თუ C მატრიცაა, ქმნის ერთ გრძელ სვეტ ვექტორს, როლის ელემენტებია C მატრიცის პიველი სვეტის ელემენტები, რომელსაც მოყვება მეორე სვეტის ელემენტები და ა. შ.

ამით არ ამოიწურება ორწერტილოვანი ოპერატორის ფუნქციები, ჩვენ მას კიდევ დავუბრუნდებით.

შესაძლებელია მატრიცის მნიშვნელობები შევიყვანოთ პირდაპირ კლავიატურიდან **input** ბრძანების საშუალებით, რომელსაც ეკრანზე გამოჰყავს ტექსტი და ელოდება მონაცემებს.

მომხმარებლის მიერ მიწოდებული მონაცემებით შეიქმნება შესაბამისი მატრიცა. თუ არ შევიყვანთ არანაირ მონაცემს (enter . . .enret), შეიქმნება ცარიელი მატრიცა. თუ ბრძანება არ მთავრდება წერტილ-მძიმით, შეყვანილი მონაცემების საფუძველზე შექმნილი მატრიცა დაიბეჭდება ბრძანებათა ფანჯარაში.

განვიხილოთ ბრძანება:

```
z = input('Enter value for z');
```

ეს ბრძანება მოგვცემს ეკრანზე ტექსტს: Enter value for z. უნდა შევიყვანოთ მონაცემები [5.1 6.3 –18.0], რომელიც მიენიჭება z. რადგან ბრძანება მთავრდება წერტილბძიმით, z მნიშვნელობა არ გამოჩნდება ეკრანზე.

სავარჯიშო

იპოვეთ ზომა და შემადგენლობა შემდეგი მატრიცებისათვის, თუ :

$$G = \begin{bmatrix} 0.6 & 1.5 & 2.3 & -0.5 \\ 8.2 & 0.5 & -0.1 & -2.0 \\ 5.7 & 8.2 & 9.0 & 1.5 \\ 0.5 & 0.5 & 2.4 & 0.5 \\ 1.2 & -2.3 & -4.5 & 0.5 \end{bmatrix}$$

11. A = G(: , 2);
12. B = G(4, :);
13. C = [10 : 15];
14. D = [4 : 9; 1 : 6];
15. E = [-5, 5];
16. F = [0.0 : 0.1 : 1.0];
17. T1 = G(4 : 5, 1 : 3);
18. T2 = G(1 :2 : 5, :);

მატრიცის დაბეჭდვა

არსებობს რამდენიმე ხერხი მატრიცის ეკრანზე გამოსაყვანად. უმარტივესია შევიყვანოთ მატრიცის სახელი და Enter. ბრძანებათა ფანჯარაში გამოჩნდება მატრიცის სახელი და მისი გამოსახულება.

როცა მატრიცის მნიშვნელობები ეკრანზე გამოგვყავს, მთელი რიცხვები გამოდის, როგორც მთელი, ხოლო ათწილადები გამოდის სპეციალური ფორმატით ე.წ. მოკლე ფორმატით - მძიმის შემდეგ 5 ნიშანი, თუ არ მივუთითებთ განსხვავებულ ფორმატს ჩვენ თვითონ. თუ გვსურს გამოვიყვანოთ რიცხვითი გამოსახულება მძიმის შემდეგ 15 ნიშნით, ვისარგებლებთ ბრძანებით **format long**. თუ კვლავ მოკლე ფორმატს გვინდა დავუბრუნდეთ – **format short.**

როცა რიცხვითი გამოსახულება ძალიან დიდია, ან ძალზე მცირე, ათწილადური ფორმა მოუხერხებელი და არაზუსტია. მაგალითად ქიმიაში ხშირად გამოიყენება ავოგადროს რიცხვი, რომლის მნიშვნელობა = 602, 300, 000, 000, 000, 000, 000, 000. მივმართავთ გამოსახულიბის ჩაწერის ეხპონენციალურ (მაჩვენებლიან) ფორმას: 1 და 10 შორის რიცხვი

გამრავლებული 10-ის შესაბამის ხარისხზე. ავოგადროს რიცხვი ასე ჩაიწერება 6.023 X 1023. 6.023 ეწოდება რიცხვის მანტისა, ხოლო 23 - ექსპონენტა. MATLAB-ში ასეთი ფორმით რიცხვის ჩაწერისას მანტისა და ექსპონენტა გამყოფილია ნიშნით e. მაგალითად 6.023 X 1023 = 6.023e+23. თუ გვინდა რიცხვითი გამოსახულება გამოვიყვანოთ ექსპონენციალური ფორმატით, ვსარგებლობთ ბრძანებით: **format short e** ან **format long e.**

ფორმატირების შემდეგი ბრძანებაა **format +.** ამ შემთხვევაში დადებითი რიცხვის მაგიერ დაიბეჭდება ნიშანი +, ისევე როგორც უარყოფითის მაგირ – ნიშანი, ხოლო თუ რიცხვის მნიშვნელობა 0 – ის ტოლია მის მაგიერ ცარიელი ადგილი დარჩება.

ტექსტის გამოყვანა ეკრანზე. ტექსტის გამოსაყვანად ეკრანზე MATLAB – ს აქვს ბრძანება **disp** , რომლის შემდეგ ფრჩხილებში იწერება ბრჭყალებში მოთავსებული ტექსტი. ეს ბრძანება ასევე გამოიყენება მატრიცის, ვექტორის ან სკალარის ეკრანზე გამოსაყვანად. მაგ. თუ სკალარი სახელით temp შეიცავს ტემპერატურის რიცხვით მნიშვნელობას-78 ფარენგეიტის გარდუსებში, ბრძანება: **disp(temp)**; **disp('degrees F')** გვაძლევს:

```
78
    degrees F
```

ფორმატირებული ბეჭდვა. ფორმატირების უფრო ვრცელ საშუალებას იძლევა ბრძანება **fprintf**. იგი ერთდროულად ტექსტის და მატრიცის დაბეჭდვის და რიცხვითი გამოსახულების ფორმატირების საშუალებას იძლევა. მისი ზოგადი ფორმა ასეთია:

fprintf (*format, matrices*)

სადაც **format** შეიცავს ბრჭყალებში ჩასმულ ტექსტს და ფორმატის სპეციფიკაციებს. ტექსტის შიგნით **%e, %f da %g** გამოიყენება იმისათვის, რომ მივუთითოთ სად უნდა დაიბეჭდოს მატრიცის რიცხვითი მნიშვნელობები. **%e** შემთხვევაში რიცხვითი გამოსახულება მოიცემა ექსპონენციალური ფორმით, **%f** შემთხვევაში რიცხვითი გამოსახულება მოიცემა ათწილადური ფორმით, **%g** შემთხვევაში – აირჩევა უმოკლესი ფორმა ექსპონენციალურსა და ათწილადურ ფორმას შორის. თუ ტექსტში არის ნიშანი **\n** ეს ნიშნავს, რომ მის მარჯვნივ მდგომი ინფორმაცია გადავა შემდეგ ხაზზე.

fprintf (' The temperature is %f degrees F \n ' , temp)

შეესაბამება:

The temperature is 78.000000 degrees F

თუ ბრძანებას ასე შევცვლით:

fprintf (' The temperature is \n %f degrees F \n ' , temp)

მივიღებთ:

```
The temperature is
78.000000 degrees F
```

ფორმატის განმსაზღვრელი ნიშნები %f, %g, %e შეიძლება შეიცავდეს ინფორმაციას თუ რამდენ ნიშანს უნდა შეიცავდეს რიცხვითი გამოსახულება და რამდენი ნიშანი დაიბეჭდოს მძიმის შემდეგ.

fprintf (‘ The temperature is %4.1f degrees F \n ’ , temp)

იძლევა:

The temperature is 78.0 degrees F

x-y გრაფიკის აგება

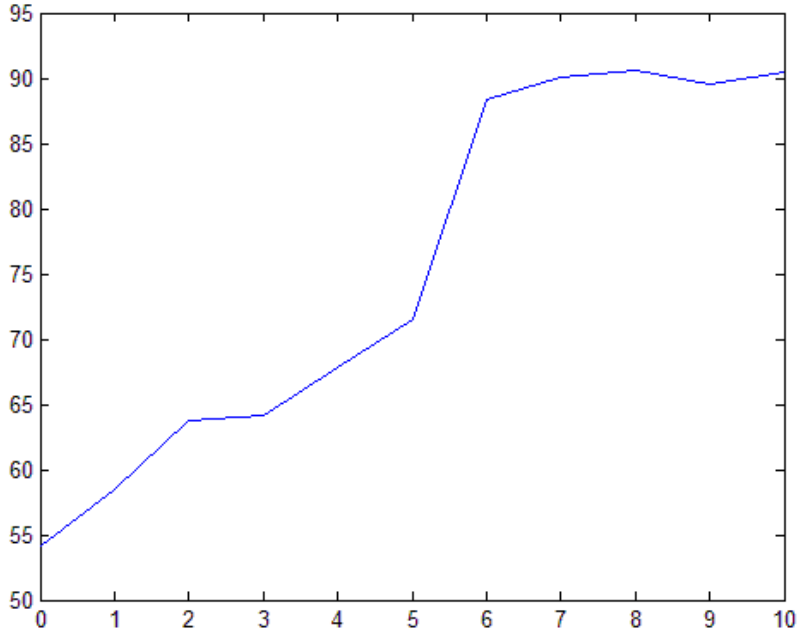
მე-7 თავში დაწვრილებით განვიხილავთ MATLAB-ის გრაფიკულ შესაძლებლობებს. ახლა კი იმის საილუსტრაციოდ თუ რაოდენ მარტივი და მოხერხებულია გრაფიკის აგება MMATLAB-ში ავაგოთ გრაფიკი x და y ვექტორების სახით წარმოდგენილი მონაცემების მიხედვით.

დავუშვათ გვინდა ავაგოთ ტემპერატურის მონაცემები, რომელიც ექსპერიმენტის ჩატარების დროს შევკრიბეთ.

დრო, წამებში	ტემპერატურა, ფარენჰეიტის გრადუსებში
0	54.2
1	58.5
2	63.8
3	64.2
4	67.8
5	71.5
6	88.3
7	90.1
8	90.6
9	89.5
10	90.4

ასევე დავუშვათ, რომ დრო მოცემული გვაქვს როგორც x ვეტორის, ხოლო ტემპერატურა – y ვექტორის ელემენტები. (ეს ვექტორები წინასწარ უნდა შევქმნათ ან ფაილიდან გამოვიძახოთ ჩვენს სამუშაო სივრცეში). გრაფიკის ასაგებად ვიყენებთ ბრძანებას **plot**, არგუმენტით x, y, სადაც ორივე ან სვეტი ვექტორია, ან სტრიქონი.

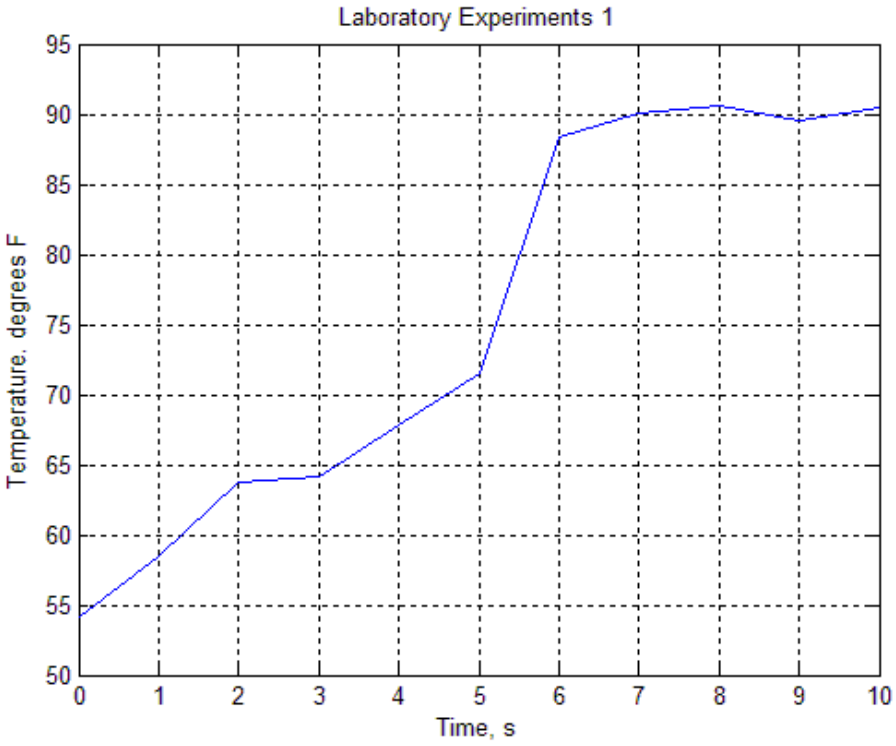
plot(x,y)



ნახ. 2.2 x, y გრაფიკი

ავტომატურად შეიქმნება გრაფიკი ნახ. 2.2. იმისათვის, რომ გრაფიკი უფრო ინფორმაციული იყოს, გავუკეთოთ მას სათაური, ღერძებს დავაწეროთ შესაბამისი სახელები და დავიტანოთ ნახაზზე საკოორდინატო ბადე ნახ. 2.3.

```
plot(x,y),. . .
title('Laboratory Experiments 1'), . . .
xlabel('Time, s'), . . .
ylabel('Temperature. degrees F'), . . .
grid
```



ნახ. 2.3 გაუმჯობესებული გრაფიკი

სამი წერტილი ყოველი სტრიქონის ბოლოს საჭიროა იმისათვის, რომ MATLAB – მა ეს ხუთივე ბრძანება ერთდროულად გაუშვას. ამის მაგივრად შეგვეძლო თითოეული ბრძანება ცალ-ცალკე მიგვეცა ერთმანეთის მიყოლებით.

თუ ერთიდაიგივე პროგრამაში გვინდა ავაგოთ ორი ნახაზი, ან ავაგოთ გრაფიკი და გავაგრძელოთ შემდგომი გამოთვლები, MATLAB გამოიყვანს ნახაზს და გააგრძელებს პროგრამაში მითითებულ ბრძანებებს, მაგრამ თუ გვინდა, რომ ნახაზს შევხედოთ, ვიდრე პროგრამა გააგრძელებს მუშაობას, **plot** ბრძანების შემდეგ უნდა დავწეროთ ბრძანება **pause**. ასეთ შემთხვევაში MATLAB დაელოდება ჩვენს დასტურს (ENTER კლავიში) პროგრამის გაგძელებისათვის.

MATLAB – ის გრაფიკულ შესაძლებლობებს დაწვრილებით გავეცნობით მე-7 თავში.

პრობლემა – გასროლილი ქვის ტრაექტორია

მცირე ზომის ობიექტი გაისოლეს დედამიწიდან 50 მილი/საათი სიჩქარით. დედამიწის ზედაპირის მიმართ 30 გრადუსი დახრის კუთხით. განვსაზღვროთ ფრენის დრო და დედამიწაზე დაცემამდე განვლილი მანძილი.

1. ამოცანის დასმა

მცირე ზომის ობიექტი გაისოლეს დედამიწიდან 50 მილი/საათი სიჩქარით. დედამიწის ზედაპირის მიმართ 30 გრადუსი დახრის კუთხით. განვსაზღვროთ ფრენის დრო და დედამიწაზე დაცემამდე განვლილი მანძილი.

საჭიროა დამატებითი ინფორმაცია:

- ობიექტის პარამეტრები და ფრენის გარემო გავლენას მოახდენს ფრენის ტრაექტორიაზე. მაგალითად ტუ ობიექტი მსუბუქია, აქვს დიდი ზედაპირული ფართი და მოძრაობს ჰაერში ჰაერის წინააღმდეგობა მნიშვნელოვნად იმოქმედეს მის რტრაექტორიაზე. გარდა ამისა, თუ ქარი ქრის, სხეული ტრაექტორია შეიცვლის. თუ ეს ინფორემაცია არ გვაქვს, უნდა დავუშვათ რომ გარემო არ მოქმედებს სხეულის გადაადგილებაზე.
- გრავიტაციული აჩქარება ასევე მოქმედებს ფრენაზე. თუ ხვა მონაცემები არ გვაქვს, ვვარაუდობთ, რომ სხეულზე მოქმედებს დედამიწის ზედაპირზე მოქმედი აჩქარება.
- უნდა შევაფასოთ საწყისი სიჩქარის და გასროლის კუთხის გაზომვის სიზუსტე, ასევე გასროლის მდებარეობაც. საიდან მოხდა გასროლა დედამოწის ზედაპირიდან თუ ადამიანის მკლავის სიმაღლიდან
- ზომის ერთეულების თანაფარდობა:
  1 მილი = 5280 ფუტი
  1 საათი = 60 წუთი = 3600 წამი
  360 გრადუსი = 2π რადიანი

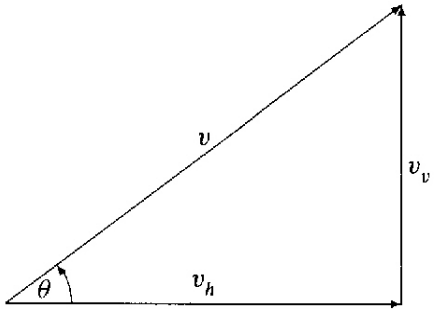
2. input/output აღწერა

- საწყისი მონაცემებია: საწყისი სიჩქარე და 50 მილი/საათში და კუთხე 30 გრადუსი ჰორიზონტალუტი მიმართულების მიმართ.
- შედეგად უნდა მივიღოთ ფრენის დრო და განვლილი მანძილი. შევარჩიოთ ზომის ერთეულებიც: წამი დროისათვის და ფუტი მანძილისათვის.

3. მათემატიკური მოდელი

შემოვიღოთ აღნიშვნები:

- დრო – t (წმ), t = 0 სხეულის გასროლის მომენტში
- საწყისი სიჩქარე = 50 მილი/საათში
- გასროლის კუთხე 30 გრადუსი
- სხეულის ჰორიზონტალური მდებარეობა x(t) (ფუტი)
- ვერტიკალური მდებარეობა - y(t) (ფუტი)
- გრავიტაციული აჩქარება g = 32.2 ft/s2, y მიმართულების საპირისპიროდ



ტრიგონომეტრიიდან ცნობილია

$V_h = V \cos \theta$
$V_v = V \sin \theta$

საწყისი სიჩქარე დავშალეთ ვერტიკალურ და ჰორიზონტალურ მდგენელებად.
წარმოვადგინოთ სიჩქარეები როგორც დროის ფუნქცია:

$$x(t) = vt\cos\theta$$

$$y(t) = vt\sin\theta - \frac{1}{2}gt^2$$

4. გამოთვლის მეთოდი

ობიექტის გასროლა ხდება როცა მისი ვერტიკალური მდებარეობა ნულის ტოლია

$$y(t) = vt\sin\theta - \frac{1}{2}gt^2$$

მას ორი ამონახსნი აქვს

$$t = 0 \qquad vt\sin\theta - \frac{1}{2}gt^2 = 0$$

მეორე ამონახსნიდან გამომდინარე ობიექტი დაეცემა დედამიწაზე დროის მომენტში:

$$t_g = \frac{2v\sin\theta}{g}$$

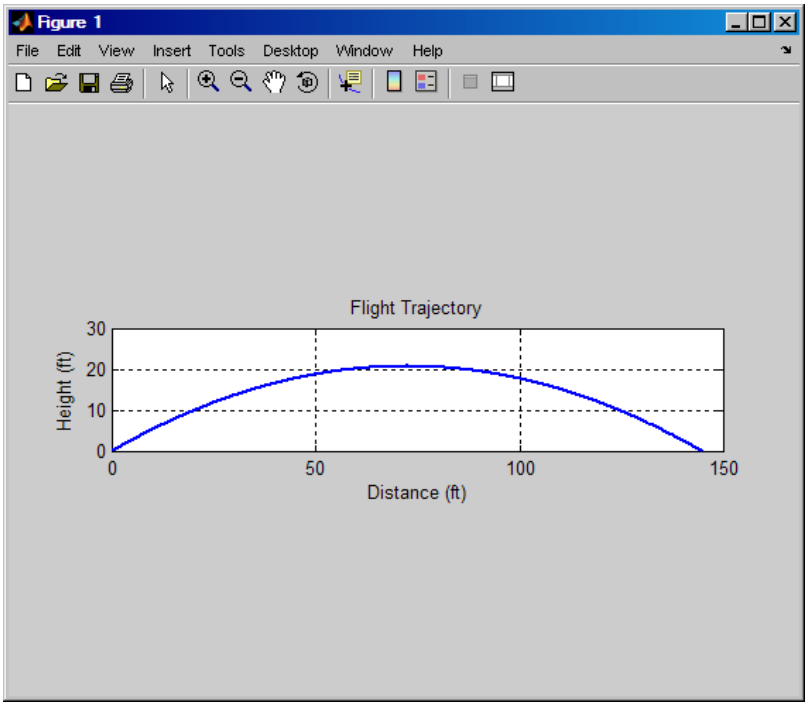
ამ დროისათვის ჰორიზონტალური მდებარეობა (განვლილი მანძილი) ტოლია:

$$x(t_g) = vt_g\cos\theta$$

5. MATLAB ამოხსნა

```
%      Flight trajectory computation
%
%     Initial values
g = 32.2; % gravity, ft/s^2
v = 50 * 5280/3600; % launch velocity, ft/s
theta = 30 * pi/180; % launch angle, radians
%     Compute and display results
disp('time of flight (s):') % label for time of flight
tg = 2 * v * sin(theta)/g % time to return to ground, s
disp('distance traveled (ft):') % label for distance
xg = v * cos(theta) * tg % distance traveled
%     Compute and plot flight trajectory
t = linspace(0,tg,256);
x = v * cos(theta) * t;
y = v * sin(theta) * t - g/2 * t.^2;
plot(x,y), axis equal, axis([ 0 150 0 30 ]), grid, ...
xlabel('Distance (ft)'), ylabel('Height (ft)'), title('Flight
Trajectory')
```

6. შემოწმება



პრობლემა – აეროდინამიკური გვირაბი

აეროდინამიკური გვირაბი – ეს არის სპეციალური შენობა სხვადასხვა სიჩქარის ქარის მოდელირებისათვის. სიჩქარის ნაცვლად შესაძლოა ვისარგებლოთ მახის რიცხვით, რომელიც

= ქარის სიჩქარე / ბგერის სიჩქარე. შენობაში თავსდება საჰაერო ხომალდის ზუსტი მოდელი და იზომება რა ძალით მოქმედებს ქარი მასზე სხვადასხვა სიჩქარისა და მოდელის მიმართ მიმართულების სხვადასხვა კუთხის შემთხვევაში. ხანგრძლივი ტესტირების შემდეგ მონაცემთა ვრცელი წყება გროვდება და მისი ანალიზის საფუძველზე დგინდება სხვადასხვა აეროდინამიკური პარამეტრი (აეროდინამიკური ამწევი ძალა-lift, აეროდინამიკური წინაღობა - drag) მოცემული მოდელისათვის.

ამ მაგალითს ჩვენ კიდევ რამდენჯერმე დავუბრუნდებით. ახლა კი დავუშვათ, რომ ტესტირებისას მიღებული მონაცემები ჩაწერილი ASCII ფაილში რომლის სახელია wind1.dat. უნდა ვნახოთ ამ მონაცემების გრაფიკი. თითოეული ხაზი ფაილში შეიცავს ფრენის მიმართულების კუთხეს გრადუსებში და შესაბამის lift კოეფიციენტს. მაგალითისათვის ავითოთ ასეთი მონაცემები:

ფრენის კუთხე (გრადუსებში)	კოეფიციენტი
-4	-0.202
-2	-0.050
0	0.108
2	0.264
4	0.422
6	0.573
8	0.727
10	0.880
12	1.027
14	1.150
15	1.195
16	1.225
17	1.224
18	1.250
19	1.245
20	1.221
21	1.177

თუმცა ვიღებთ მონაცემთა მხოლოდ მცირე ნაწილს, მაგრამ ესეც საკმარისია დავრწმუნდეთ მონაცემთა ფაილის გამოყენების უპირატესობაში. ფაილში ჩაწერილი მონაცემები მრავალგზის შეგვიძლია გამოვიყენოთ სხვადასხვა ამოცანებისათვის.

წარმოგიდგენთ ხუთ საფეხურიან პროცესს იმის საილუსტრაციოდ რომ ეს მარტივი პროცესია.

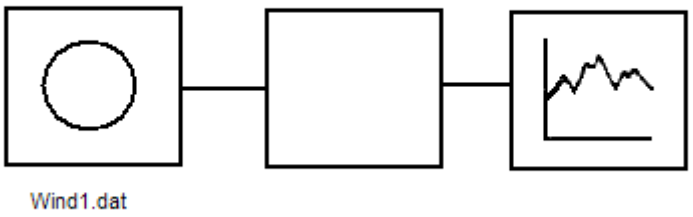
1. ამოცანის დასმა

ავაგოთ ფრენის მიმართულების კუთხის გრაფიკი კოეფიციენტის მიმართ.

2. input/output აღწერა

სადაც ეს შესაძლებელია ვისარგებლებთ I/O დიაგრამით, როგორც ეს ნაჩვენებია ნახ. 2.4. ამ მაგალითში ვკითხულობთ ინფორმაციას ფაილიდან და MATLAB საშუალებით ვაგებთ

გრაფიკს. დიაგრამა შეიცავს სიმბოლოს დისკეტისათვის, რაც წარმოადგენს მინაცემთა ფაილს – input. output არის გრაფიკული გამოსახულება და მისთვის ვიყენებთ სხვა აღნიშვნას.



ნახ. 2.4 I/O დიაგრამა

3. სახელდახელო ამოხსნა

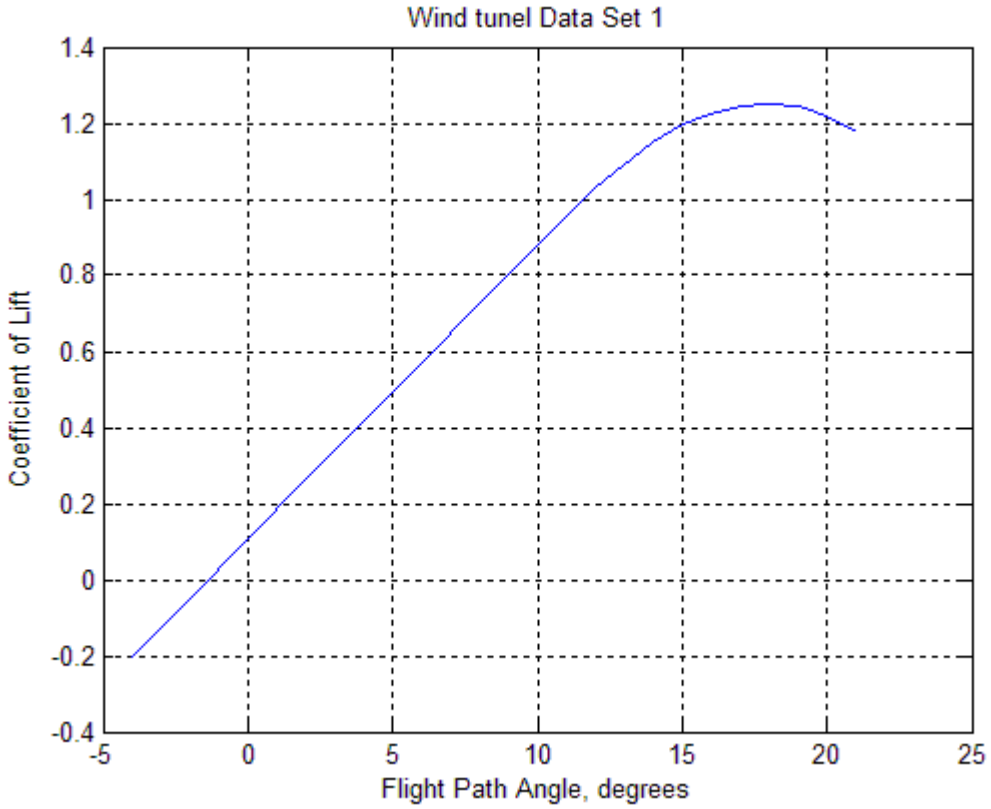
იმისათვის, რომ დაახლოებით წარმოდგენა გვქონდეს, რა შედეგი უნდა მივიღოთ, შევასრულოთ ამოცანა მცირე მასშტაბით. ამ შემთხვევაში თუ დავხედავთ მონაცემებს ვნახავთ, რომ კოეფიციეფიციენტი იზრდება –0.2 დან და აღწევს მაქსიმუმს 1.25, 18 გრადუსზე.

4. MATLAB ამოხსნა

```
%
%     This program reads and plots information
%     in data file containing flight path angles
%     and coefficients of lift.
%
load wind1.dat;      % read data from file to matrix
x = wind1(:,1);      % copy first column into x vector
y = wind1(:,2);      % copy second column into y vector
plot(x,y),...        % plot x and y
title('Wind tunel Data Set 1'),...
xlabel('Flight Path Angle, degrees'),...
ylabel('Coefficient of Lift'),...
grid
```

5. შემოწმება

თუ გავუშვებთ ამ პროგრამას MATLAB –ში მივიღებღებთ გრაფიკს. ნახ. 2.5.



ნახ. 2.5 აეროდინამიკური ექსპერიმენტის მონაცემთა გრაფიკი

ამ თავში ზოგადად წარმოგიდგინეთ MATLAB – ის გარემო. მისი ძირითადი სტრუქტურული ერთეულია მატრიცა, რომელიც შეიძლება იყოს სკალარი – ჩვეულებრივი რიცხვი, ვექტორი – რიცხვითი მწკრივი, ან რიცხვთა სტრიქონებად და სვეტებად განლაგებული ერთობლიობა. გაეცანით მატრიცის შექმნის მეთოდებს, ASCII ან MAT ფაილის სახით ჩაწერილი მონაცემების გამოძახებას, გაჩვენეთ როგორ უნდა აიგოს x-y გრაფიკი. თავის დასასულს ავაგეთ აეროდინამიკური ტესტირების შედეგად მიღებული მონაცემების გრაფიკი.

ჩამოთვლილია ყველა სპეციალური სიმბოლო, ბრძანება და ფუნქცია, რომელიც განხილული იყო ამ თავში. თითოეულ მათგანს ერთვის მოკლე განმარტება.

სპეცსიმბოლოები:

[ ]	გამოიყენება მატრიცის შესაქმნელად
( )	მატრიცის ელემენტების იდექსებისათვის
,	გამოყოფს ერთმანეთისაგან ინდექსებსა და მატრიცის ელემენტებს
;	გამოყოფს მატრიცებს ან ბრძანებებს, კრძალავს ბეჭდვას
>>	მოგვიხმობს შემდეგი ბრძანებისაკენ
. . .	აგრძელებს ბრძანებას შემდეგ ხაზზე
%	განსაზღვრავს კომენტარს ან ფორმატს
:	მატრიცის შესაქმნელად
\n	განსაზღვრავს ახალ ხაზს
^C	ბრძანების ლოკალურად შეწყვეტა

ბრძანებები და ფუნქციები

ans	ცვლადი, თუ მას შექმნისას სახელს არ მივანიჭებთ
clc	ასუფთავებს ბრძანებების ფანჯარას
clear	ასუფთავებს სამუშაო სივრცეს
clf	ასუფთავებს გრაფიკულ ფანჯარას
computer	გვიჩვენებს კომპიუტერის ტიპს
demo	იწყებს MATLAB დემონსტრირებას
disp	გამოყავს ეკრანზე მატრიცა ან ტექსტი
echo	ეკრანზე გამოგვიყვანს M ფაილის შინაარსს
exit	შეზყვეტს MATLAB - ს
format +	გამოყავს მხოლოდ ნიშანი და არა რიცხვითი გამოსახულება
format long	ათწილადი გამოყავს 15 ნიშნით მძიმის შემდეგ
format short	ათწილადი გამოყავს 5 ნიშნით მძიმის შემდეგ
format long e	რიცხვი გამოყავს ექსპონენციალური ფორმით
format short e	რიცხვი გამოყავს ექსპონენციალური ფორმით
fprintf	ბეჭდავს ფორმატირებულ ტექსტს
grid	გრაფიკულ გამოსახულებაზე გამოჰყავს საკოორდინატო ბადე
help	გამოიძახბს MATLAB help
input	საშუალებას გვაძლევს მონაცემი შევიყვანოთ კლავიატურიდან
load	გამოიძახებს მატრიცას ფაილიდან
pause	დროებით წყვეტს პროგრამას
plot	აგებს გრაფიკს
quit	წყვეტს MATLAB
save	ინახავს ცვლადს ფაილში
size	განსაზღვრავს მატრიცის ზომას
title	გრაფიკული გამოსახულების სათაური
type	დაგვიბეჭდავს ფაილის შინაარსს
what	ჩამოგვითვლის M ფაილებს მოც. სამუშაო სივრცეში
who	ჩამოთვლის ცვლადებს მოც. სამუშაო სივრცეში

whos	ჩამოთვლის ცვლადებს მოც. სამუშაო სივრცეში + ზომები
xlabel	გრაფიკზე x ღერძის სახელი
ylabel	გრაფიკზე y ღერძის სახელი

ამოცანები

1 – 10 ამოცანაში გამოიყენეთ ორწერტილოვანი ოპერატორი, რომ შექმნათ მითითებული ვექტორი და შემდეგ დაბეჭდეთ შესაბამისი ცხრილი თქვენ არ გჭირდებათ არცერთი არითმეტუკული ოპერაცია MATLAB-ში ამ პრომლემების გადასაწყვეტად, მაგრამ შესაძლოა დაგჭირდეთ ნაზრდის გამოთვლა.

1. შექმენით გრადუსების რადიანებში გადასაყვანი ცხრილი. პირველი ხაზი უნდა შიცავდეს სიდიდეებს 0 გრადუსისთვის, მეორე ხაზი – 10 გრადუსისთვის და ა. შ. ბოლო ხაზი უნდა შეიცავდეს სიდიდეებს 360 გრადუსისათვის.
2. შექმენით რადიანების გრადუსებში გადასაყვანი ცხრილი. დაიწყეთ რადიანების სვეტი 0.0 ნაზრდით $\pi/10$ 2π-მდე. (შეგახსენებთ $\pi=180^0$).
3. შექმენით დუიმების სანტიმეტრებში გადასაყვანი ცხრილი. დაიწყეთ დუიმების სვეტი 0.0 ნაზრდით 0.5, ბოლო სტრიქონი უნდა შეიცავდეს მონაცემს 20 დუიმი (1 დუიმი = 2.54 სმ).
4. შექმენით სანტიმეტრების დუიმებში გადასაყვანი ცხრილი. დაიწყეთ სანტიმეტრების სვეტი 0.0 ნაზრდით 0.5, ბოლო სტრიქონი უნდა შეიცავდეს მონაცემს 50 სმ. (1 დუიმი = 2.54 სმ).
5. შექმენით მილი/საათების ფუტი/წმ.-ში გადასაყვანი ცხრილი. დაიწყე მილი/საათების სვეტი 0.0 ნაზრდით 5 მილი/საათი ბოლო სტრიქონი უნდა შეიცავდეს მონაცემს 65 მილი/საათი (1 მილი = 5.280 ფუტს)
6. შექმენით ფუტი/წმ. მილი/საათებში გადასაყვანი ცხრილი. დაიწყეთ ფუტი/წმ. სვეტი 0.0 ნაზრდით 5 ფუტი/წმ. ბოლო სტრიქონი უნდა შეიცავდეს მონაცემს თქვენს მიერ შერჩეულ სიდიდეს (1 მილი = 5.280 ფუტს)

შემდეგი პირობები ვრცელდება 7 – 10 ამოცანებზე:

$ = 5.3 ფრანკი

1 იენი = 0.0079 $

1.57 გერმანული მარკა = 1 $

7. შექმენით ფრანკების დოლარებში გადასაყვანი ცხრილი. დაიწყე ფრანკების სვეტი 5ფრანკით, ნაზრდით 5 ფრანკი. დაბეჭდეთ ასეთი ცხრილის 25 სტრიქონი.
8. შექმენით გერმანული მარკის ფრანკში გადასაყვანი ცხრილი ცხრილი. დაიწყე მარკების სვეტი 1გერმანული მარკით, ნაზრდით 5 გ.მ. მიეცი საშუალება მომხმარებელს თვითონ შეიყვანოს სტრიქონების რიცხვი კლავიატურიდან.
9. შექმენით იენის გერმანულ მარკებში გადასაყვანი ცხრილი. დაიწყე იენის სვეტი 100 იენით და დაბეჭდეთ ასეთი ცხრილის 25 სტრიქონი, ისე, რომ იენის საბოლოო მნიშვნელობა იყოს 10 000.

10. შექმენით დოლარის ფრანკებში, გერმანულ მარკაში და იენში გადასაყვანი ცხრილი. დაიწყე დოლარის სვეტი 1 დოლარით. დაბეჭდეთ ასეთი ცხრილის 50 სტრიქონი.

11-15 ამოცანები მომართავენ მონაცემთა ASCII ფაილს სახელით temp.dat . მონაცემთა ფაილი შეიცავს 100 სტრიქონს. თითოეული სტრიქონი შეიცავს ახალი ძრავის ტესტირების შედეგებს. პირველი სიდიდე სტრიქონში(0.0) არის დრო, როცა ძრავამ მუშაობა დაიწყო. დროის ყოველ მოცემულ მომენტში იზომებოდა ტემპერატურა ძრავის 4 სხვადასხვა ადგილას. ამრიგად, ყოველი სტრიქონი შეიცავს 5 მონაცემს: დრო და 4 ტემპერატურა (T1, T2, T3, T4).

11. ააგეთ გარფიკი, სადაც x ღერძზე გადაზომილია დრო, y ღერძზე კი T1. დააწერეთ ღერძებს შესაბამისი აღნიშვნები.
12. ააგეთ გარფიკი, სადაც x ღერძზე გადაზომილია დრო, y ღერძზე კი T2. დააწერეთ ღერძებს შესაბამისი აღნიშვნები.
13. ააგეთ გარფიკი, სადაც x ღერძზე გადაზომილია დრო, y ღერძზე კი T3. დააწერეთ ღერძებს შესაბამისი აღნიშვნები.
14. ააგეთ გარფიკი, სადაც x ღერძზე გადაზომილია დრო, y ღერძზე კი T4. დააწერეთ ღერძებს შესაბამისი აღნიშვნები.
15. ააგეთ გარფიკი, სადაც x ღერძზე გადაზომილია T4, y ღერძზე კი დრო.

კოსმოსური თანამგზავრი VIKING 1

ფუნდამენტური საინჟინრო გამოთვლები

წარმოგიდგენთ მასალას, რომელიც ფუნდამენტურია ძირითადი საინჟინრო ამოცანებისათვის. მაგალითად MATLAB-ის ფუნქციები ტრიგონომეტრიული, ლოგარითმული და ექსპონენციალური გამოთვლებისათვის. ასევე განვიხილავთ კომპლექსურ რიცხვებს, რომელიც განსაკუთრებით მნიშვნელოვანია ელექტრო და მექანიკურ საინჟინრო დისციპლინებში. გარდა იმისა, რომ ვაწარმოოთ რიცხვითი ოპერაციები, უნდა შეგვეძლოს შედარების, ლოგიკური და ციკლის ოპერაციების ჩართვა პროგრამაში. ამიტომ თავში კონტროლის ოპერაციების შესახებ განხილულია საკითხი იმის შესახებ თუ როგორ უნდა დაისვას შეკითხვა პროგრამაში და პასუხზე დაყრდნობით როგორ უნდა შესრულდეს გარკვეული ოპერაციები. თავი სტსტისტიკური განაზომებისა და მათი ინტერპრეტაციის შესახებ ეხება ექსპერიმენტული შედეგების ანალიზს. განვიხილავთ შემთხვევითი სიდიდის გენერირების ფუნქციებს, რომლებიც დაგვჭირდება გარკვეული ფიზიკური პროცესების მოდელირებისათვის. MATLAB-ს ასევე გააჩნია დამატებითი ფუნქციები და ოპერტორები, რომლებიც მატრიცებისთვისაა განკუთვნილი. და ბოლოს, იმის გამო, რომ საინჟინრო კონცეფციათა არსის გაგება ბევრადაა დამოკიდებული მათ ვიზუალურ წარმოდგენაზე, ძალზე მნიშვნელოვანია მარტივად და მოხერხებულად შევძლოთ მათი წარმოჩენა. MATLAB–ს გააჩნია მძლავრი საშუალებები გრაფიკული ინტერპრეტაციისათვის.

3 სკალარული და მატრიცული გამოთვლები

შესავალი

შეკრების, გამოკლების, გამრავლების და გაყოფის ოპერაციები ძირითადი არითმეტიკული ოპერაციებია, რომელოთაც ყველაზე ხშირად ვსარგებლობთ. ასევე ხშირად ვასრულებთ ისეთ ოპერაციებს, როგორიცაა კვადრატული ფესვის ამოღება, რიცხვის ლოგარითმის ან კუთხის ტრიგონომეტრიული ფუნქციის გამოთვლა. შესაძლებელია ეს ოპერაციები შესრულდეს როგორც ერთი სიდიდისათვის ასევე მათი ერთობლიობისათვისაც – ვექტორის და მატრიცისათვის. ამ თავში გავეცნობით, როგორ ახორციელებს MATLAB ამ ოპერაციებს. გარდა ამისა, ვნახავთ, როგორ გამოვიყენოთ MATLB–ის შესაძლებლობები კომპლექსური რიცხვებისათვის. ერთ-ერთ პრაქტიკულ მაგალითში განვიხილავთ საკომუნიკაციო სიგნალის ექოს, მეორე კი ეხება ჰიდროაკუსტიკურ სიგნალს.

3.1 MATLAB სპეციალური სიდიდეები და მატრიცები

MATLAB-ს აქვს გარკვეული რაოდენობის წინასწარგანსაზღვრული მუდმივები, სპეციალური სიდიდეები და სპეციალური მატრიცები. მათი უმრავლესობა შექმნილია MATLAB-ის მიერ თავისივე ფუნქციების გამოყენებით. ამ თავში გაგაცნობთ MATLAB ფუნქციათა რამდენიმე მაგალითს. ხოლო 3.4 განყოფილებაში მიმოვიხილავთ MATLAB-ის დამატებითი ფუნქციების რამდენიმე მაგალითს.

3.1.1 სპეციალური სიდიდეები

pi	MATLAB ამ სახელიან ცვლადს აღიქვამს როგორც ჩვეულბრივ რიცხვს π.
i, J	გამოიყენება ლომპლექსული რიცხვებისათვის და MATLAB-ი მათ აღიქვამს როგორც √-1
inf	ეს ცვლადი აღნიშნავს უსასრულობას, რომელიც, ჩვეულბრივ, გამოთვლებისას მიიღება როგორც ნულზე გაყოფის შედეგი (∞)
NaN	წარმოადგენს სიდიდეს, რომელიც არ არის რიცხვი – განუზღვრელობა, სისდიდე, რომელიც მიიღება მაგალითად ნულის გაყოფით ნულზე.
clock	გვაძლევს მიმდინარე დროს ექვსელემენტიანი ვექტორის სახით: წელი, თვე, საათი, წუთი და წამი.
date	გვაძლევს მიმდინარე თარიღს სიმბოლოთა მწკრივის სახით მაგ. 20-Jun-2006
eps	მოცემული კომპიუტერის სიზუსტე (რიცხვის მოცემის) მცოცავი მძიმის რეჟიმში. ეს არის სხვაობა 1.0- სა და მის მომდევნო უახლოეს ათწილადს შორის, რომლის წარმოება მოც. კომპიუტერს შეუძლია.
ans	გამოიყენება იმ ცვლადის აღსანიშნავად, რომელის სახელიც განსაზღვრული არ არის.

3.1.2 სპეციალური მატრიცები

MATLAB აქვს ფუნქციათა ჯგუფი, რომლის საშუალებით სპეციალური ტიპის მატრიცები იქმნება, რომელთაც გარკვეული დანიშნულება აქვთ რიცხვითი მეთოდების გამოყენებაში. ზოგირთ მათგანს კი უფრო ზოგადი გამოყენება აქვს.

მაგიური კვადრატი. n რიგის მაგიური კვადრატი არის n×n მატრიცა, რომლს ელემენტებია მთელი რიცხვები 1 დან n^2 – მდე. მისი ელემენტები ისეა განლაგებული, რომ ყოველ სტრიქონში და სვეტში ელემენტების ჯამი ერთიდაიგივე რიცხვია.

ბრძანება **zeros** აწარმოებს მატრიცას, რომლის ყველა ელემენტი ნულის ტოლია. თუ ფუნქციის არგუმენტი სკალარია, მიიღება კვადრეტული მატრიცა, რომლის სვეტების და სტრიქონების რიცხვი არგუმენტით განისაღზღვრება. თუ ფუნქციას ორი სკალარული არგუმენტი ქვს, (**zeros**(m,n)), მიღება n სტრიქონიანი და m სვეტიანი მატრიცა, რომლის ყველა ელემენტი ნულის ტოლია. თუ არგუმენტი მატრიცაა, მაშინ შეიქმნება ისეთივე ზომის მატრიცა, როგორიც არგუმენტია. შემდეგი ბრძანებები შეესაბამება აღწერილ შემთხვევებს:

```
A = zeros(3);
B = zeros(3,2);
C = [ 1 2 3; 4 5 6];
D = zeros( size(C));
```

$$A = \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \qquad B = \begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix}$$

$$C = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{matrix} \qquad D = \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$$

ფუნქცია **ones** აწარმოებს მატრიცას, რომლის ყველა ელემენტი 1-ის ტოლია, ისევე, როგორც **zeros** აწარმოებს 0 –ების შემცველ მატრიცას. თუ ფუნქციის არგუმენტი სკალარია, მიიღება კვადრეტული მატრიცა, რომლის სვეტების და სტრიქონების რიცხვი არგუმენტით განისაღზღვრება. თუ ფუნქციას ორი სკალარული არგუმენტი ქვს, (**ones**(m,n)), მიღება n სტრიქონიანი და m სვეტიანი მატრიცა, რომლის ყველა ელემენტი ერთის ტოლია. თუ არგუმენტი მატრიცაა, მაშინ შეიქმნება ისეთივე ზომის მატრიცა, როგოროც არგუმენტია. შემდეგი ბრძანებები შეესაბამება აღწერილ შემთხვევებს:

```
A = ones(3);
B = ones(3,2);
C = [ 1 2 3; 4 5 6];
D = ones(size(C));
```

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \qquad D = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

ერთეულოვანი მატრიცა ისეთი მატრიცაა, რომლის მთავარი დიაგონალის ელემენტები 1-ის ტოლია, ყველა დანარჩენი კი ნულის. მაგ. შემდეგი მატრიცა არის 4×4 ერთეულოვანი მატრიცა:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

აღვნიშნავთ, რომ მთავარი დიაგონალი არის ის დიაგონალი, რომლის ელემენტების ინდექსებიც ერთმანეთის ტოლია – (1,1), (2,2), (3,3) და ა.შ.

MATLAB – ში ერთეულოვანი მატრიცა შეგვიძლია შევქმნათ **eye** ფუნქციის საშუალებით. ამ ფუნქციის არგუმენტი მსგავსია **ones** და **zeros** ფუნქციათა არგუმენტების. თუ ფუნქციის არგუმენტი სკალარია, მიიღება კვადრეტული მატრიცა, რომლის სვეტების და სტრიქონების რაოდენობა არგუმენტით განისაღზღვრება. თუ ფუნქციას ორი სკალარული არგუმენტი ქვს, (**eye**(m,n)), მიიღება n სტრიქონიანი და m სვეტიანი მატრიცა, რომლის მთავარი დიაგონალის ელემენტები ერთის ტოლია, ყველა დანარჩენი კი ნულის. თუ არგუმენტი მატრიცაა, მაშინ შეიქმნება იგივე ზომის ერთეულოვანი მატრიცა. შემდეგი ბრძანებები შეესაბამება აღწერილ შემთხვევებს:

```
A = eye(3);
B = eye(3,2);
C = [ 1 2 3; 4 5 6];
D = eye( C );
```

$$A = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad B = \begin{matrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{matrix}$$

$$C = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{matrix} \qquad D = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{matrix}$$

სასურველია ერთეულოვან მატრიცას არასოდეს დავარქვათ i, ამან შესაძლოა გამოიწვიოს პრობლემები თუ იგივე პროგრამაში კომპლექსურ რიცხვებს იყენებთ.

პასკალის სამკუთხედი. ბრძანება **pascal** აწარმოებს კვადრატულ მატრიცას, რომლის ელემენტები პასკალის სამკუთხედს შეესაბამება. პასკალის სამკუთხედი გვაძლევს ბინომიალური გაფართოების კოეფიციენტებს ფორმით: $(a + b)^n$. მაგალითად მეოთხე ხარისხის სამკუთხედი:

$$\begin{matrix} & & & 1 & & & \\ & & 1 & & 1 & & \\ & 1 & & 2 & & 1 & \\ 1 & & 3 & & 3 & & 1 \\ 1 & & 4 & & 6 & & 4 & & 1 \end{matrix}$$

თუ მივცემთ ბრძანებას **pascal**(5), მივიღებთ მატრიცას:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 6 & 10 & 15 \\ 1 & 4 & 10 & 20 & 35 \\ 1 & 5 & 15 & 35 & 70 \end{bmatrix}$$

თუ კარგად დააკვირდებით ამ მატრიცაში ადვილად იპოვით პასკალის სამკუთხედს.

3.2 სკალარული ოპერაციები

არითმეტიკული ოპერაციები ხორციელდება გამოსახულებათა საშუალებით. გამოსახულება შესაძლოა იყოს მარტივი, როგორიცაა მაგალითად მუდმივა - კონსტანტა, ანდა შეიცავდეს მატრიცებს და კონსტანტებს, შეერთებულს არითმეტიკული ოპერაციების ნიშნებით. ამ თავში განვიხილავთ მხოლოდ სკალარული სიდიდეების შემცველ გამოსახულებებს.

ცხრილი აღწერს არითმეტიკულ ოპერაციებს საკალარებს შორის.

ოპერაცია	ალგებრული ფორმა	MATLAB
შეკრება	a + b	a + b
გამოკლება	a - b	a - b
გამრავლება	a x b	a * b
მარჯვენა გაყოფა	a/b	a/b
მარცხენა გაყოფა	b/a	a\b
ახარისხება	a^b	a^b

გამოსახულება: x = a + b, a და b მნიშვნელობები შეიკრიბება და მიღებული შედეგი მიენიჭება x –ს. ხოლო გამოსახულება count = count + 1 – ნიშნავს რომ count მნიშვნელობას მიემატება 1 და შედეგი ჩაიწერწბა ისევ count ცვლადში, შეცვლის რა მის წინა მნიშვნელობას, ე. ი. count მნიშვნელობა 1 –ით გაიზრდება.

მნიშვნელოვანია აღინიშნოს, რომ ცვლადს არ შეიძლება ჰქონდეს ერთდროულად ორი მნიშვნელობა. მაგალითად ერთმანეთის მომდევნო ეს ორი ბრძანება:

time = 0.0;
time = 5.0;

შედეგად გვაძლევს time = 5.0.

3.1.3 არითმეტიკული ოპერაციების შესრულების რიგი

რამდენადაც რამდენიმე არითმეტიკული ოპერაცია შეიძლება გაერთიანდეს ერთი გამოსახულების სახით, მნიშვნელოვანია დავიცვათ მათი შესრულების რიგი. ცხრილი გვიჩვენებს არითმეტიკული ოპერაციების რიგითობას MATLB გამოსახულებაში, რაც ისეთივეა, როგორც ჩვეულებრივ ალგებრული ოპერაციების რიგი.

პროცედურა	ოპერაცია
1	ფრჩხილები
2	ახარისხება მარცხნიდან მარჯვნივ
3	გამრავლება და გაყოფა, მარცხნიდან მარჯვნივ
4	შეკრება – გამოკლება, მარცხნიდან მარჯვნივ

დავუშვათ გვსურს გამოვთვალოთ ტრაპეციის ფართობი, სადაც სკალარი height შეიცავს სიმაღლის სიგრძეს, ხოლო სკალარები base_1 და base_2 - მის ორ ფუძეს. ტრაპეციის ფართობი შეგვიძლია გამოვთვალოთ შემდეგი ბრძანებებით:

area = 0.5* height *(base_1 + base_2);

თუ გამოგვრჩა ფრჩხილები:

area = 0.5* height *base_1 + base_2;
მაშინ ეს ბრძანება იდენტური იქნება ბრძანების:

area = (0.5* height *base_1) + base_2;
და შედეგიც შესაბამისად არასწორი მოჰყვება, თუმცა პროგრამა ვერ განსაზღვრავს რომ შეცდომაა დაშვებული და არც მიგვითითებს ამის თაობაზე, ამიტომ პროგრამის შედგენისას სიფრთხილე უნდა გამოვიჩინოთ, რათა უზრუნველვყოთ ატითმეტიკული მქმედებების სათანადო რიგი. განვიხილოთ განტოლება:

$$f = \frac{x^3 - 2x^2 + x - 6.3}{x^2 + 0.05005x - 3.14}$$

f მნიშვნელობა უნდა გამოვითვალოთ შემდეგი ბრძანებებით:

numerator = x^3 – 2*n^2 + x – 6.3;
denumerator = x^2 + 0.05005 * x – 3.14;
f = numerator/denumerator;

უკეთესია ბრძანებები ასეთნაირად გავყოთ, ვიდრე ერთი გრძელი გამისახულება დავწეროთ, რომელიც მათ გააერთიანებს. ეს უფრო მარტივიცაა და შეცდომებსაც აგვაცილებს თავიდან

სავარჯიშო

დაწერეთ ბრძანებები შემდეგი გამოსახულებების გამოსათვლელად:

1. ხახუნის კოეფიციენტი საბურავსა და ქვაფენილს შორის:
$$\text{friction} = \frac{v^2}{30s}$$
2. correction faqtor(შესწორების კოეფიციენტი) წნევის გამოთვლისას:
$$factor = 1 + \frac{b}{\upsilon} + \frac{c}{\upsilon^2}$$
3. ორ წერტილს შორის დახრა:

$$slope = \frac{y_2 - y_1}{x_2 - x_1}$$

4. პარალელური წრედის წინაღობა:

$$resis\tan ce = \frac{1}{\frac{1}{r_1} + \frac{1}{r_2} + \frac{1}{r_3}}$$

3.1.4 შეზღუდვები გამოთვლებში

კომპიუტერში ცვლადმა შეიძლება მიიღოს რიცხვითი მნიშვნელობა ფართო დიაპაზონში, ხშირ შემთხვევაში ეს არის 10^{-308} – 10^{308} შუალედი. უმრავლეს შემთხვევაში ეს საკმარისია. თუმცა ზოგჯერ შედეგმა შესაძლოა გადააჭარბოს აღნიშნულ ზღვარს. დავუშვათ გვაქვს:

```
x = 2.5e200;
y = 1.0e200;
z = x*y;
```

x და y ზემოაღნიშნულ შუალედში ხვდებიან, მაგრამ z მას აჭარბებს.
ასეთ შეცდომას უწოდებენ ექსპონენციალურ გადაკავებას (overflow). MATLAB ასეთ ცვლადს მიანიჭებს მნიშვნელობას ∞.
ასევე შეიძლება მივიღოთ ძალიან მცირე რიცხვი – ექსპონენციალური უკმარისობა (underflow),

```
x = 2.5e-200;
y = 1.0e200;
z = x/y;
```

ამ შემთხვევაში z სიდიდე მიიღებს ნულოვან მნიშვნელობას.

3.3 ოპერაციები მასივებზე

მასივებზე ოპერაცია ხორციელდება შესაბამის ელემენტებს შორის. მაგალითად, დავუშვათ A არის ხუთელემენტიანი სტრიქონი ვექტორი. B ასევე ხუთელემენტიანი სტრიქონი ვექტორია. იმისათვის, რომ შეიქმნას ვექტორი, რომლის ელემენტებიც ამ ორი ვექტორის ელემენტების ნამრავლი იქნება, ერთი გზა ასეთია:

```
C(1) = A(1) * B (1);
C(2) = A(2) * B (2);
C(3) = A(3) * B (3);
C(4) = A(4) * B (4);
C(5) = A(5) * B (5);
```

ეს ოპერაციები სკალარულია, რადგან შეიცავს მხოლოდ სკალარულ სიდიდეებს.

თუ გვსურს ორი ერთნაირი ზომის ვექტორი ან მატრიცა ერთმანეთზე წევრ-წევრად გადავამრავლოთ, მოქმედების ნიშნის წინ უნდა დავწეროთ წერტილი:

```
C = A.*B
```

წერტილის გამოტოვება ამ ოპერაციაში სერიოზული შეცდომაა, რადგან ასეთ შემთხვევაში მოხდება მატრიცული გადამრავლება, რომელიც როგორც ვიცით, სრულიადაც არ ნიშნავს ელემენტების ერთმანეთზე წევრ-წევრად გადამრავლებას.

შეკრებისა და გამოკლებისათვის მატრიცული და მასივებზე ოპერაციები ერთიდაიგივეა, ხოლო გამრავლების გაყოფის, ახარისხების მატრიცული ოპერაციები განსხვავდება მასივებზე ოპერაციებისაგან, ამიტომ არ უნდა დაგვავიწყდეს მოქმედების ნიშნის წინ წერტილის დასმა, როცა ეს საჭიროა (მასივებზე ოპერაციის დროს). ეს წესები შეჯამებულია ცხრილში:

ოპერაციები შესაბამის ელემენტებზე

ოპერაცია	ალგებრული ფორმა	MATLAB
შეკრება	a + b	a + b
გამრავლება	a – b	a - b
გამრავლება	a x b	a. * b
მარჯვენა გაყოფა	a/b	a./b
მარცხენა გაყოფა	b/a	a.\b
ახარისხება	a^b	a.^b

```
B = 3*A; - თუ   A სკალარია,
B = 3.*A; - თუ   A არ არის სკალარი.
```

შედეგად მიღებული ვექტორი იგივე ზომოს იქნება, როგორიც მოქმედებაში მონაწილე ვექტორები.

```
A = [2 5 6]
B = [2 3 5]
```

A ,B მასივური ნამრავლი:

```
C = A. *B  = [4 15 30].
```

MATLAB აქვს გაყოფის ორგვარი ოპერაცია: მარჯვენა და მარცხენა. (/, \).

C = A./B – მარჯვენა გაყოფა ხორციელდება A მატრიცის ელემენტების გაყოფით B-მატრიცის შესაბამის ელემენტებზე. ხოლო მარცხნიდან გაყოფა – B eლემენტების გაყოფით A ელემენტებზე.

მასივური ახარისხებაც შესაბამისად წევრ-წევრად ხდება. მაგალითად:

```
C = A.^2;
D = A.^B;
```

შესაბამისად მოგვცემს:

```
C = [4   25  36]
D = [4   125   7776]
```

შეგვიძლია ახარისხებისას ფუძედ სკალარი ავიღოთ, მაჩვენებლად – ვექტორი:

```
C = 3.0.^ A – მოგვცემს C = [9 243 729].
```

ექვივალენტურია ბრძანება:

```
C = (3).^A;
```

ამგვარად დიდი სიფრთხილეა საჭირო რომ სწორად განვსაზღვროთ და მივუთითოთ პროგრამაში რა ტიპის ოპერაცია გვჭირდება მატრიცული თუ მასივური.

მოყვანილ მაგალითებში ვიხილავდით ვექტორებს. ყოველივე რაც მათ შესახებ ითქვა, ეხება მატრიცებსაც.

```
d = [1:5; -1 : –1 : -5];
z = ones(d);
s = d – z;
p = d.*s;
sq = d. ^3;
```

მიიღება:

$$d = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ -1 & -2 & -3 & -4 & -5 \end{bmatrix} \qquad z = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$s = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ -2 & -3 & -4 & -5 & -6 \end{bmatrix} \qquad p = \begin{bmatrix} 0 & 2 & 6 & 12 & 20 \\ 2 & 6 & 12 & 20 & 30 \end{bmatrix}$$

$$sq = \begin{bmatrix} 1 & 8 & 27 & 64 & 125 \\ -1 & -8 & -27 & -64 & -125 \end{bmatrix}$$

სავარჯიშო

გამოითვალეთ C ვექტორის მნიშვნელობები, თუ A = [2 –1 5 0] B = [3 2 –1 4]

1. C = A- B;
2. C = B +A – 3;
3. C = 2*A + A.^B;
4. C = B. /A;
5. C = B. \A;
6. C = A. ^ B;
7. C = (2).^B +A;
8. C = 2*B/3.0.*A;

ამოცანა - საკომუნიკაციო სიგნალის ექო

შესრულებული იქნა საინტერესო გამოკვლევა კომპიუტერული სისტემის განვითარებისათვის, რომელიც ემყარება ხმოვან ბრძანებებს. ასეთი სისტემა გულისხმობს, რომ მიკროფონი, რომლითაც ბრძანებები გადაიცემა რაც შეიძლება ზუსტად გადასცემდეს მეტყველებას. სამწუხაროდ სენსორებს, მათ შორის მიკროფონსაც ახასიათებს ხმაური. ორმხრივი კომუნიკაციის სისტემებს ახასიათებს აგრეთვე ექო. ვიდრე შეუდგებოდეს სიტყვების

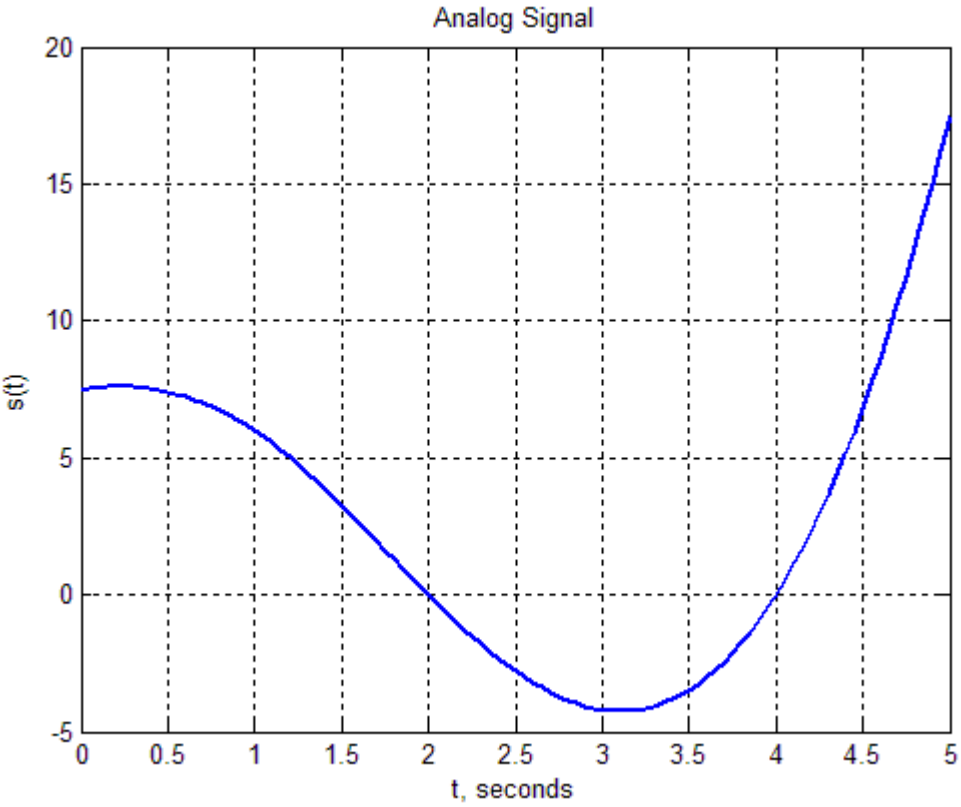
გაიგივებას, მეტყველების გამაიგივებელ სისტემას უნდა შეეძლოს გადაამუშაოს სიგნალი და გაასუფთაოს იგი არასასურველი კომპონენტებისაგან, როგორიცაა მაგალითად ექო. იმისათვის, რომ შევამოწმოთ პროგრამა, რომელმაც უნდა გაასუფთაოს სიგნალი ექოსაგან, გვჭირდება შევქმნათ ციფრული სიგნალი და მას დავურთოთ ექოები. შემდეგ მიღებულ სიგნალს დავამუშავებთ გამოსაცდელი პროგრამის საშუალებით და ვნახავთ რამდენად კარგად შევძელით დასახული ამოცანის შესრულება. განვსაზღვროთ ციფრული სიგნალი და დავწეროთ MATLAB პროგრამა, რომელიც მას დაუმატებს რამდენიმე ექოს.

ციფრული სიგნალი

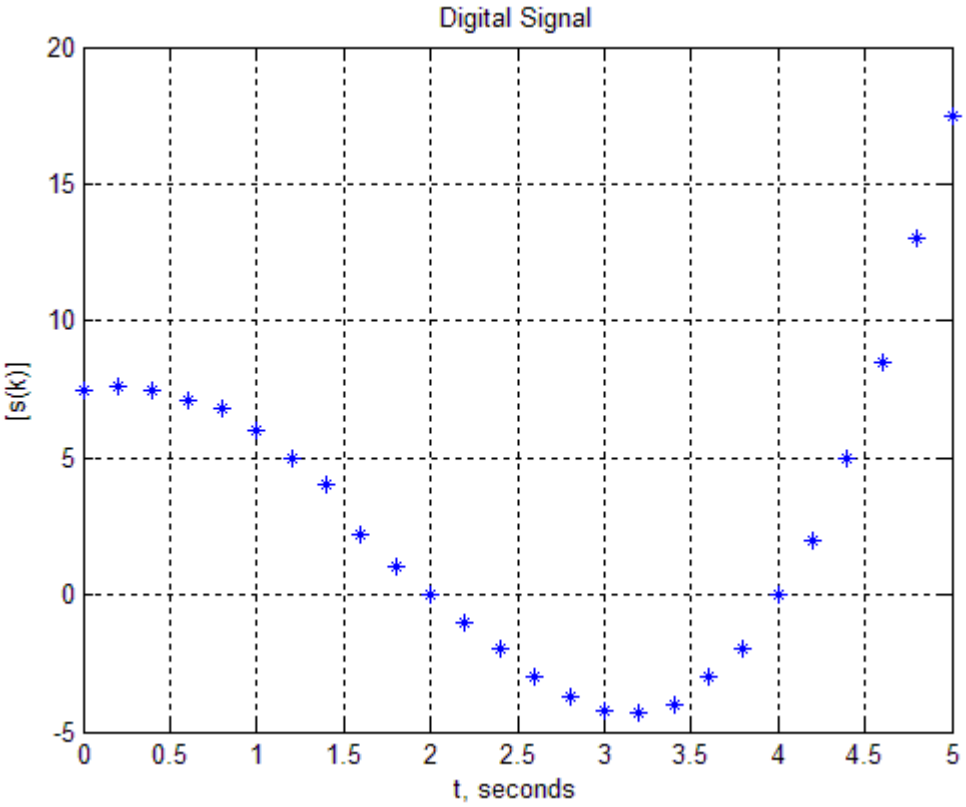
სიგნალი არის ფუნქცია (ჩვეულებრივ დროისა) რომელიც გვაწვდის ინფორმაციას. ინფორმაცია, ანუ მონაცემები შეკრებილია სენსორის მიერ. ჩვეულებრივ ასეთი მიმღებია მიკროფონი, რომელიც ზომავს აკუსტიკურ ან ხმოვან სიდიდეებს(მაგალითად მეტყველება), სეისმომეტრი, რომელიც ზომავს დედამიწის ქერქის მოძრაობას, ფოტომიმღები, რომელიც ზომავს სინათლის ინტენსივობას, თერმომეტრი- ტემპერატურას, ვოლტმეტრი – ძაბვას. სენსორი ჩვეულებრივ მიერთებულია სხვა ინსტრუმენტთან, ე. წ. ანალოგურ-ციფრულ გარდამქმნელთან(A/D), რომელიც აგროვებს სიგნალს პერიოდულად, ჩაიწერს დროითი ანათვალს და სიგნალის შესაბამის მნიშვნელობას მონაცემთა ფაილის სახით. საწყისი, შემომავალი სიგნალი ჩვეულებრივ უწყვეტი (ანალოგური) ფუნქციაა. ამგვარად მიღებულ მონაცემთა მწკრივს ციფრულ სიგნალს უწოდებენ. ნახ 3.1 წარმოადგენს უწყვეტი სიგნალის მაგალითს, 3.2 - მონაცემთა მწკრივს, რომელიც მიიღება ანალოგური სიგნალიდან - ციფრულ სიგნალს. ციფრული სიგნალი შეიცავს მონაცემთა მწკრივს, რომელიც შეიძლება ჩაიწეროს ფაილის სახით და შემდეგ გამოვიყენოთ MATLAB –ში დაწერილ პროგრამაში, ავაგოთ გრაფიკი სიგნალის მნიშვნელობებისა დროის მიმართ. გრაფიკის აგებისას წერტილებს ერთმანეთთან ვაერთებთ სეგმენტებით, რომ შედეგი უფრო თვალსაჩინო იყოს.

ექოს გენერირება

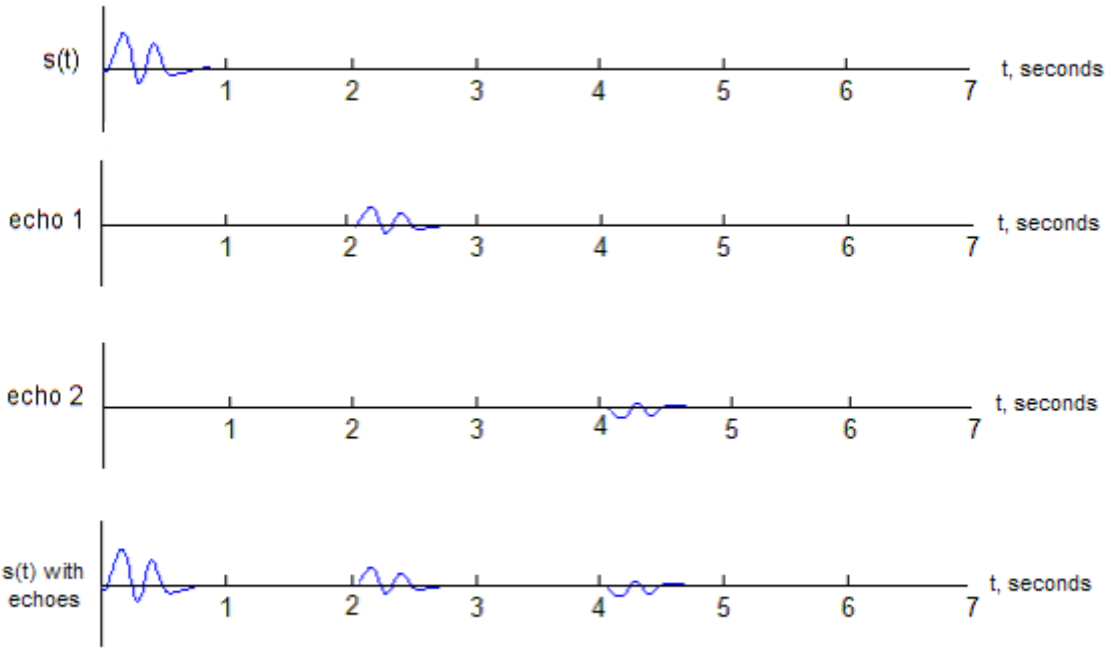
ექო განიხილება როგორც შესუსტებული ანარეკლი ორიგინალური სიგნალისა, რომელიც სიგნალს დაერთვის გარკვეული შეყოვნებით. მაგალითად ნახ 3.3 შეიცავს ორიგინალურ სიგნალს, s(t), მეორე გრაფაში სიგნალის მნიშვნელობა შესუსტებულია 0.5 ჯერ და 2 წამის დაგვიანებით დაერთვის სიგნალს. მესამე გრაფაში საიგნალს ერთვის ექო, რომელიც შესუსტებულია 0.3 ჯერ და დაგვიანებული 5 წამით. ეს ე.წ. ჩახვეული (rolled) ექოა,რადგან ექოს მნიშვნელობები უარყოფითი სიდიდეებია. მეოთხე გრაფაში წარმოდგენილია ორიგინალური სიგნალი თანდართული ორივე ექოთი.



ნახ. 3.1 ანალოგური სიგნალი



ნახ. 3.2 ციფრული სიგნალი



ნახ. 3.3 სიგნალი ექოებითურთ

დავუშვათ მონაცემები სიგნალის შესახებ გროვდებოდა 10 წამის განმავლობაში, ყოველ 0.1 წამში. პირველი წამის განმავლობაში შეგროვდა კოორდინატთა შემდეგი წყება:

დრო, წმ	სიგნალის მნიშვნელობა
00	0.0
0.1	0.5
0.2	1.0
0.3	-0.5
0.4	0.75
0.5	0.0
0.6	-0.02
0.7	-0.10
0.8	0.0
0.9	0.0
1.0	0.0

სიგნალის შემდგომი მნიშვნელობები 0-ის ტოლია.

დაწერეთ MATLAB პროგრამა, რომელიც შექმნის სიგნალს, რომელიც შედგება საწყისი სიგნალისა და დართული სამი ექოსაგან. პირველი ექო – სიგნალი შესუსტებულია ფაქტორით - 0.5 და დაგვიანებული 2 წამით, მეორე ექო – ჩახვეული(rolled) ექო 4 წამის დაგვიანებით და შესუსტებული ფაქტორით -0.3, მესამე ექო დაგვიანებულია 7.5 წამით და შესუსტებული ფაქტორით 0.1. ააგეთ საწყისი სიგნალი და ახალი სიგნალი სამივე ექოთი. შემდეგ შეინახეთ ორივე სიგნალის მონაცემები M ფაილის სახით – echo.m.

1. ამოცანის დასმა

მოცემულია საწყისი სიგნალი, შევქმნათ ახალი სიგნალი, რომელიც შეიცავს როგოც საწყის სიგნალს, ასევე სამ სხვადასხვაგვარად შესუსტებულ და დაგვიანებულ ექოს.

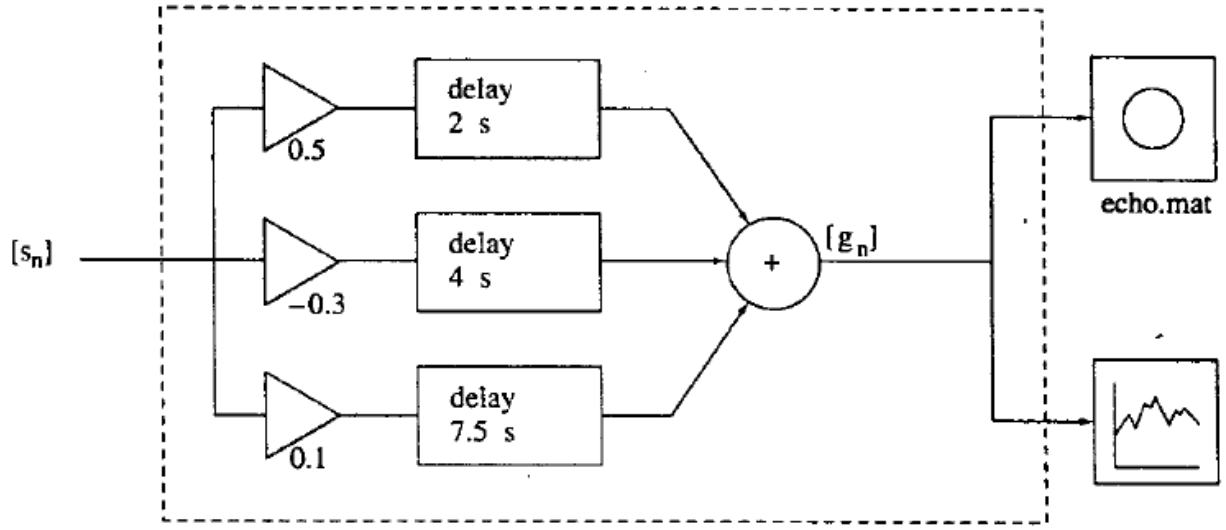
2. INPUT/OUTPUT აღწერა

ნახ. 3.4 –ის პუნქტირით შემოსაზღვრული არე მოიცავს საწყისი სიგნალისგან (s_n) ექოს გენერირების დაწვრილებით სურათს. სიგნალის მნიშვნელობები გამრავლებულია განსაზღვრულ კოეფიციენტზე, რომ მივიღოთ ექოს შესაბამისი მნიშვნელობები, შემდეგ საწყისი სიგნალი და სამივე ექო შეკრებილია ახალი სიგნალის სახით (g_n), რომელიც უნდა ავაგოთ და შევინახოთ ფაილში echo.mat.

3. სახელდახელო ამოხსნა

ავიღოთ პირველი 3 მონაცემი:

დრო, წმ	სიგნალის მნიშვნელობა
00	0.0
0.1	0.5
0.2	1.0



ნახ. 3.4 I/O დიაგრამა

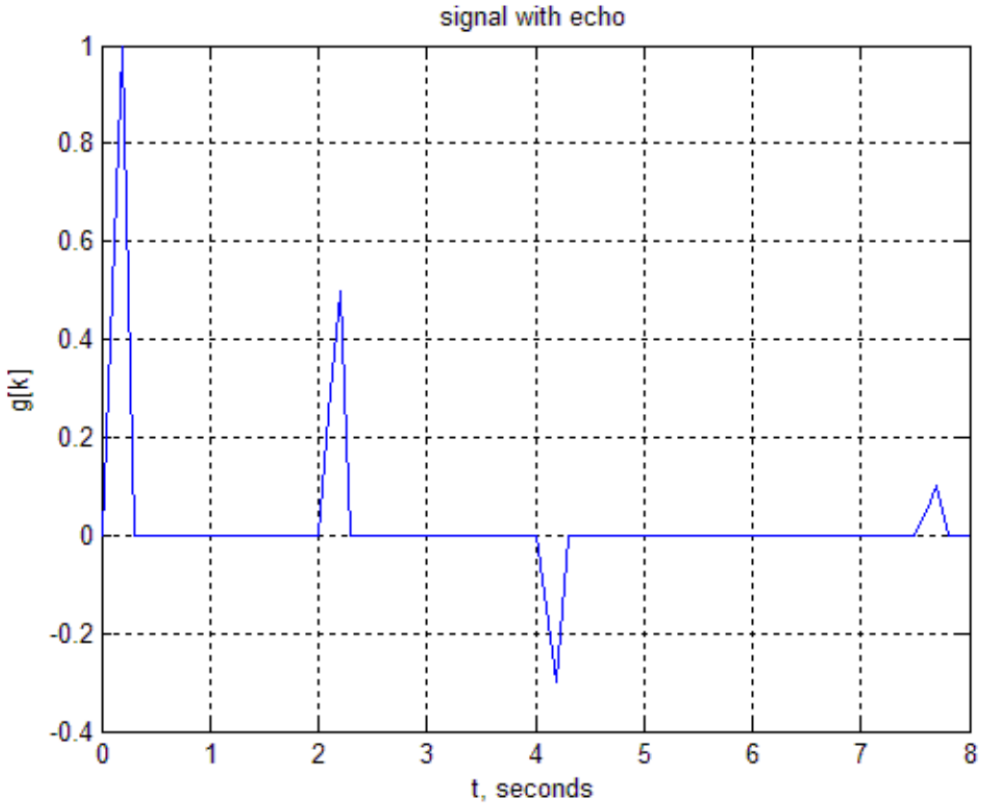
მივიღებთ:

2.0 (0.5)*(0.0) = 0.0
2.1 (0.5)*(0.5) = 0.25
2.2 (0.5)*(1.0) = 0.5

4.1 (-0.3)*(0.5) = -0.15
4.2 (-0.3)*(1.0) = -0.3

7.5 (0.1)*(0.0) = 0.0
7.6 (0.1)*(0.5) = 0.05
7.5 (0.1)*(1.0) = 0.1

ნახაზზე 3.5 მოცემულია საწყისი სიგნალი + სამი ექოს გრაფიკი



ნახ. 3.5 საწყისი სიგნალი 3 ექოთი

4. MATLAB ამოხსნა

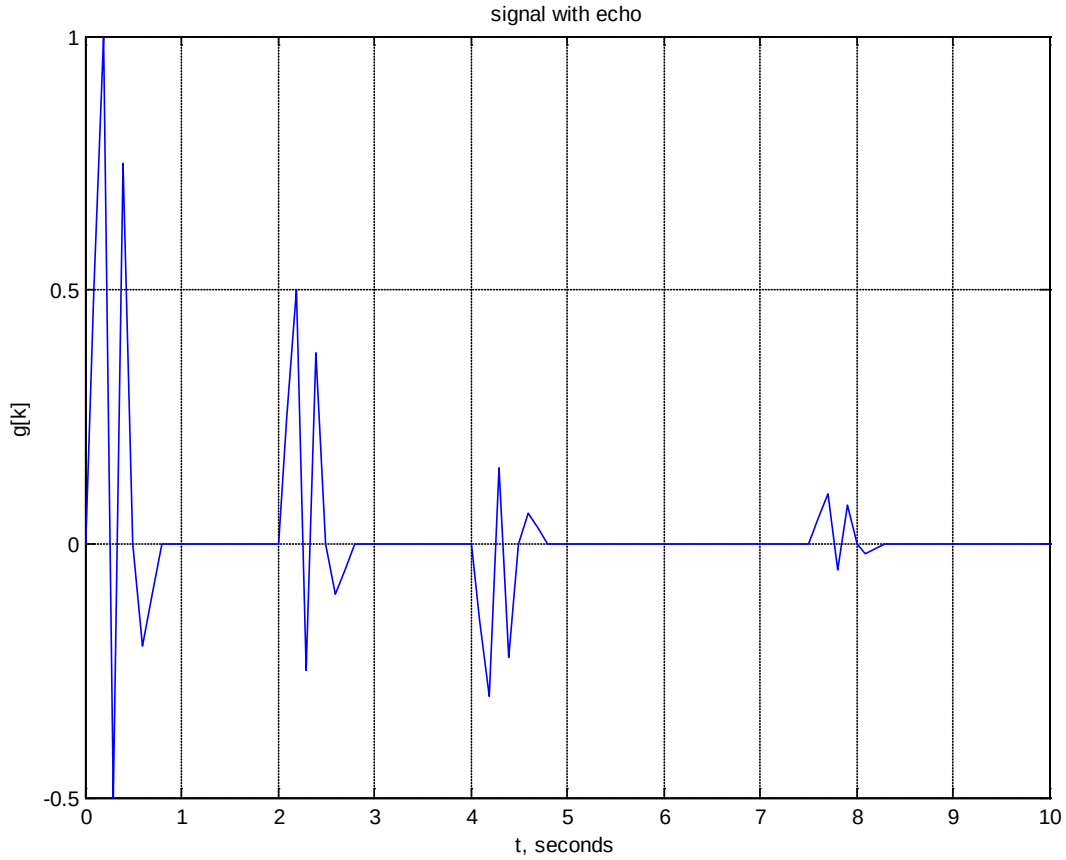
იმისათვის, რომ გამოვითვალოთ ინდექსები ექოსათვის, გავიხსენოთ ფაქტი, რომ ათვლის ინტერვალია 0.1. ამიტომ იმისათვის, რომ შევქმნათ ექო 2 წამის დაგვიანებით უნდა დავაგვიანოთ ექო ათვლის 20 წერტილით. s(1) მნიშვნელობამ უნდა შექმნას echo_1(21) მნიშვნელობა, s(2) – echo_1(22) და ა.შ. MATLAB პროგრამა ასე გამოიყურება:

```
%              This program generates tree echoes from an
%               original signal
```

```
%               The sum of the original signal and the tree
%                echoes is stored
%                in a data file and then plotted.

t=0.0:0.1:10;
s=zeros(size(t));
s(1:8)=[0 0.5 1 -0.5 0.75 0 -0.2 -0.1];
%
%          Generate three echoes by scaling and
%          delaiing  original signal.
%
echo_1=zeros(size(t));
echo_1(21:101)=0.5*s(1:81);
echo_2=zeros(size(t));
echo_2(41:101)=-0.3*s(1:61);
echo_3=zeros(size(t));
echo_3(76:101)=0.1*s(1:26);
%
%           Add echoes to original signal giving new signal.

%          Save time signal and new signal.
%
g=s+echo_1+echo_2+echo_3;
save echo3 t g;
%
%             plot new signal.
%
plot(t,g),...
    title('signal with echo'),...
xlabel('t, seconds'),...
ylabel('g[k]'),...
grid
```

5. შემოწმება

გრაფიკი 3.6 შეიცავს საწყის სიგნალს და სამ ექოს.



ნახ. 3.6 საწყისი სიგნალი 3 ექოთი

3.4 ძირითადი მათემატიკური ფუნქციები

გარდა მარტივი არითმეტიკული ოპერაციებისა საჭიროა სხვა ძირითადი მათემატიკური ოპერაციები როგორიცაა ლოგარითმი, ტრიგონომეტრიული ფუნქციები, და სხვა. მათთვის MATLAB გააჩნია ფუნქციები. მაგალითად თუ გვინდა გამოვითვალოთ კუთხის სინუსი, ვიყენებთ შესაბამის ფუნქციას:

```
b = sin(angle);
```

იგულისხმება რომ **sin** ფუნქციის არგუმენტი რადიანებშია გამოსახული. თუ არგუმენტი გრადუსებშია მოცემული, იგი წინასწარ უნდა გადავიყვანოთ რადიანებში:

```
b = sin(angle*pi/180);
```

შეგვიძლია კუთხე წინასწარ გადავიყვანოთ რადიანებში:

```
angle_radians = angle*pi/180;
b = sin(angle_radians);
```

ეს ბრძანებები ვარგისია იმ შემთხვევაშიც, როცა კუთხე სკალარია, და მაშინაც, როცა კუთხის მნიშვნელობები ვექტორის სახითაა მოცემული. თუ კუთხის ნიშვნელობები მატრიცის სახითაა მოცემული, მაშინ sin დაითვლება თითოეული ელემენტისათვის.

განვიხილოთ წესები, რომელიც MATLAB –ის ფუნქციებს შეეხება. ფუნქციას გააჩნია არგუმენტი - პარამეტრები, რომელიც ფრჩხილებში იწერება ფუნქციის სახელის შემდეგ. ფუნქციას შესაძლოა არც გააჩნდეს არგუმენტი, ან გააჩნდეს ერთი ან რამდენიმე არგუმენტი. მაგალითად pi ფუნქციაა, რომელსაც არგუმენტი არ აქვს. როცა ჩვენ მივუთითებთ pi იგი ავტომატურად გვაძლევს π მნიშვნელობას. თუ ფუნქციას გააჩნია ერთი ან რამდენიმე არგუმენტი, ძალიან მნიშვნელოვანია არგუმენრების მიმდევრობის დაცვა. ზოგიერთი ფუნქცია მოითხოვს, რომ არგუმენტი მოცემული იყოს განსაზღვრულ ერთეულში. მაგალითად, ტრიგონომეტრიული ფუნქციების არგუმენტის განზომილება უნდა იყოს რადიანი. ზოგიერთ ფუნქციას შესაძლოა ჰქონდეს არგუმენტების ცვალებადი რაოდენობა იმის მიხედვით, თუ რა გვსურს შედეგად მივიღოთ. ასე მაგალითად, ფუნქციას **zeros** შეიძლება ჰქონდეს ერთი ან ორი არგუმენტი რაც განსაზღვრავს შედეგას.

არ შეიძლება ფუნქციას მოჰყვებოდეს ტოლობის ნიშანი, რადგან იგი ფუნქციაა და არა ცვლადი, რომელმაც რაღაც მნიშვნელობა უნდა მიიღოს, მაგრამ დასაშვებია, რომ ფუნქცია თავად მოჰყვებოდეს ტოლობის ნიშანს. შესაძლებელია, აგრეთვე, რომ რაიმე ფუნქცია სხვა ფუნქციის არგუმენტის ნაწილი იყოს. მაგალითად, შემდეგი ბრძანება გამოითვლის x ცვლადის აბსოლუტური სიდიდის ლოგარითმს:

```
log_x = log(abs(x));
```

როცა ერთი ფუნქცია გამოიყენება იმისათვის, რომ გამოვითვალოთ სხვა ფუნქციის არგუმენტი, კარგად დააკვირდით, რომ არგუმენტები შესაბამისი ფუნქციების მისთვის განკუთვნილ ფრჩხილებში იქნას ჩაწერილი.

განვიხილოთ ფუნქციათა ის კატეგორია, რომელიც ყველაზე ხშირად გამოიყენება საინჟინრო ამოცანეების პროგრამირებისას.

3.1.5 ელემენტარული მათემატიკური ფუნქციები.

ეს ფუნქციები საჭიროა იმისათვის, რომ შევასრულოთ ზოგადი გამოთვლები,როგორიცაა ვიპოვოთ აბსოლუტური სიდიდე ან გამოვთვალოთ კვადრატული ფესვი. MATLAB –ს გააჩნია რამდენიმე ფუნქცია რიცხვითი სიდიდის დამრგვალებისათვის:

abs(x)	გამოთვლის x ცვლადის აბსოლუტურ სიდიდეს
sqrt(x)	ამოიღებს კვადრატულ ფესვს x სიდიდიდან
round(x)	ამრგვალებს x სიდიდეს უახლოეს მთელამდე
fix(x)	ამრგვალებს x სიდიდეს უახლოეს მთელამდე 0 მიმართულებით
floor(x)	ამრგვალებს x სიდიდეს უახლოეს მთელამდე -∝მიმართულებით
ceil(x)	ამრგვალებს x სიდიდეს უახლოეს მთელამდე +∝ მიმართულებით
sign(x)	გვაძლევს –1, თუ x<0, 0 – თუ x=0, 1 – თუ x>0
rem(x,y)	გვაძლევს x სიდიდის y –ზე გაყოფის შედეგად მიღებულ ნაშთს
exp(x)	გვაძლევს e^x, სადაც e ნატურალური ლოგარითმის ფუძეა (ნეპერის რიცხვია) ~ 2.718282
log(x)	გვაძლევს x-ის ლოგარითმს ნატურალური ფუძით
log10(x)	გვაძლევს x-ის ლოგარითმს 10-ის ფუძით

სავარჯიშო

განსაზღვრეთ შემდეგი გამოსახულებები და პასუხები შეამოწმეთ MATLAB საშუალებით

1. round(-2.6)
2. fix(-2.6)
3. floor(-2.6)
4. ceil(-2.6)
5. sign(-2.6)
6. abs(round(-2.6))
7. sqrt(floor(10.7))
8. rem(15,2)
9. floor(ceil(10.8))
10. log10(100) + log10(0.001)
11. abs(-5:5)
12. round([0:0.3:2, 1:0.75:4])

ტრიგონომეტრიული ფუნქციებით სარგებლობისას უნდა გვახსოვდეს, MATLAB არგუმენტს აღიქვამს რადიანებში. იმისათვის, რომ რადიანი გადავიყვანოთ გრადუსებში ან პირიქით, გამოვიყენოთ შემდეგი გამოსახულებები, ვიცით რა $180^o = \pi$ რადიანს:

```
angle_degrees = angle_radians*(180/pi);
angle_radians = angle_degrees*(pi/180);
```

sin(x)	გამოითვლის x ცვლადის სინუსს, x გამოსახულია რადიანებში
cos(x)	გამოითვლის x ცვლადის კოსინუსს, x გამოსახულია რადიანებში
tan(x)	გამოითვლის x ცვლადის ტანგენსს, x გამოსახულია რადიანებში
asin(x)	გამოითვლის x ცვლადის არკსინუსს, ანუ შებრუნებულ სინუსს,სადაც x მდებარეობს –1 და 1 შორის. ფუნქცია გვაძლევს კუთხეს $-\pi/2$ და $\pi/2$ შორის.
acos(x)	გამოითვლის x ცვლადის არკკოსინუსს, ანუ შებრუნებულ კოსინუსს,სადაც x მდებარეობს –1 და 1 შორის. ფუნქცია გვაძლევს კუთხეს $-\pi/2$ და $\pi/2$ შორის.
atan2(x,y)	გამოითვლის არკტანგენსს, ანუ შებრუნებულ ტანგენსს სიდიდისა y/x. ფუნქცია იძლევა კუთხეს რადიანებში, რომელიც მდებარეობს საზღვრებში $-\pi + \pi$, x და y ნიშნების მიხედვით.

სხვა ტრიგონომეტრიული ფუნქციები შეიძლება გამოვითვალოთ:

$$\sec x = \frac{1}{\cos x}$$

$$\csc x = \frac{1}{\sin x}$$

$$\cot x = \frac{1}{\tan x}$$

სავარჯიშო

მიეცით MATLABS ბრძანებები იმისათვის, რომ გამოითვალოთ შემდეგი სიდიდეები, იმ დაშვებით, რომ განტოლებებში შემავალი ცვლადები სკალარული სიდიდეებია და წინასწარ ცნობილია მათი რიცხვითი მნიშვნელობები.

1. თანაბრად აჩქარებული მოძრაობა

$$motion = \sqrt{\upsilon\iota^2 + 2 \cdot a \cdot x}$$

2. ელექტრული რხევის სიხშირე:

$$frequency = \frac{1}{\sqrt{\frac{2\pi \cdot c}{L}}}$$

3. ტყვიის მოძრაობის ტრაექტორია:

$$range = 2\upsilon i^2 \cdot \frac{\sin(b) \cdot \cos(b)}{g}$$

4. length contraction

$$length = k\sqrt{1 - \left(\frac{\upsilon}{c}\right)^2}$$

5. fillet რგოლის მოცულობა

$$volume = 2\pi x^2 \left(\left(1 - \frac{\pi}{4}\right) \cdot y - \left(0.833 - \frac{\pi}{4}\right) \cdot x\right)$$

6. მანძილი მიზიდულობის ცენტრიდან წაკვეთილი ცილინდრის კვეთის სიბრტყიდან (distance of the center of gravity from a reference plane in a hollow cylinder sector):

$$center = \frac{38.1972 \cdot \left(r^3 - s^3\right)\sin\alpha}{\left(r^2 - s^2\right) \cdot \alpha}$$

ჰიპერბოლური ფუნქცია წარმოადგენს ნატურალური ხარისხის (e^x) ფუნქციას. შებრუნებული ჰიპერბოლური ფუნქცია კი x სიდიდის ნატურალური ფუძით ლოგარითმია (ln x). ეს ფუნქციები გამოიყენება ისეთ ამოცანებში, როგორიცაა მაგალითად ზოგიერთი ტიპის ციფრული ფილტრის დიზაინი. MARLAB - ში არსებობს ცპეციალური ფუნქციები მათ გამოსათვლელეად.

sinh(x)	გამითვლის x ცვლადის ჰიპერბოლურ სინუსს, რომელიც ტოლია $\frac{e^x - e^{-x}}{2}$
cosh(x)	გამითვლის x ცვლადის ჰიპრბოლურ კოსინუსს რომელიც ტოლია $\frac{e^x + e^{-x}}{2}$
tanh(x)	გამითვლის x ცვლადის ჰიპერბოლურ ტანგენსს, x რომელიც ტოლია $\frac{\sinh x}{\cosh x}$
asinh(x)	გამოითვლის x ცვლადის შებრუნებულ ჰიპერბოლურ სინუსს -

$\ln\left(x+\sqrt{x^2+1}\right)$

acosh(x) გამოითვლის x ცვლადის შებრუნებულ ჰიპერბოლურ კოსინუსს - $\ln\left(x+\sqrt{x^2-1}\right)$, x >= 1-თვის

atanh(x) გამოითვლის x სიდიდის შებრუნებულ ჰიპერბოლურ ტანგენსს- $\frac{1}{2}\ln\left(\frac{1+x}{1-x}\right)$ |x| < 1-თვის.

სხვა ჰიპერბოლური ფუნქციები გამოითვლება შემდეგი ფორმულების საშუალებით:

$$\coth x = \frac{\cosh x}{\sinh x} \qquad x \neq 0$$

$$\sec hx = \frac{1}{\cosh x}$$

$$\csc hx = \frac{1}{\sinh x}$$

$$a\coth x = \frac{1}{2}\ln\left(\frac{x+1}{x-1}\right) \quad |x| > 1$$

$$a\sec hx = \ln\left(\frac{1+\sqrt{1-x^2}}{x}\right) \quad 0 < x < 1$$

$$aschx = \ln\left(\frac{1}{x}+\frac{\sqrt{1+x^2}}{|x|}\right) \quad x \neq 0$$

სავარჯიშო

გამოითვალეთ MATLAB საშუალებით შემდეგი გამოსახულებები. x მნიშვნელობა წინასწარ ცნობილია ან შედეგია უკვე ჩატარებული გამოთვლებისა.

1. coth x
2. sec x
3. csc x
4. acoth x
5. asech x
6. acsc x

3.1.6 M ფაილები – MATLAB –ის ახალი ფუნქციები

ზოგიერთი სახის გამოთვლები, რომლებიც უფრო ხშირად გჭირდებათ, შეგიძლიათ აქციოთ MATLAB ფუნქციებად. ამისათვის უნდა დაიწეროს პროგრამა, რომელიც საჭირო გამოთვლებს ჩაატარებს და ჩაიწეროს იგი M ფაილის სახით. შეგიძლიათ ეს ფუნქცია გამოიყენოთ, როგორც MATLAB სხვა ფუნქციები. ასეთი ფაილები გარკვეული წესების დაცვით უნდა დაიწეროს. ვიდრე განვიხილავთ ამ წესებს, მოვიყვანთ M-file ფუნქციის მაგალითს, რომელიც ჩაწერილია circum.m ფაილის სახით:

```
function c = circum ( r )
%  CIRCUM        Circumference of circle with radius r.
%                For matrices, CIRCUM ( r ) returns a matrix
%                containing the circumferences of circles
%                with radii equal to the values in the original vector.
c = pi*2*r;
```

ეს ფუნქცია შეგვიძლია გამოვიყენოთ r რადიუსიანი წრეწირის სიგრძის გმოსათვლელად

```
r = [ 0 1.4 pi ];
circum( r );
```

MATLAB ახალი ფუნქციის შექმნისას უნდა დავიცვათ შემდეგი წესები:

1. პირველი სტრიქონი უნდა დაიწყოს სიტყვით function, რომელსაც მოჰყვება შედეგად მისაღები არგუმენტები, ტოლობის ნიშანი და ფუნქციის სახელი. ფუნქციის საწყისი არგუმენტები იწერება ფრჩხილებში ფუნქციის სახელის შემდეგ. ეს სტრიქონი განსაზღვრავს შესავალ და გამოსავალ არგუმენტებს (input – output) და იგი განასხვავებს ასეთ ფაილს სხვა m ფაილებისაგან.
2. შემდეგი რამდენიმე ხაზი უნდა დაეთმოს კომენტარს, რომელიც ეკრანზე გამოვა, თუ MATLAB მივცემთ ბრძანებას **help** circum.
3. შედეგად მიღებული მნიშვნელობები მიენიჭება გამოსავალ არგუმენტს, ამიტომ შეამოწმეთ: ფუნქციის ტექსტი უნდა შეიცავდეს ბრძანებას, რომელიც გამოსავალ არგუმენს მიანიჭებს გამოთვლილ მნიშვნელობას.
4. ფუნქციასა და პროგრამაში, რომელიც მას იყენებს მატრიცებისთვის ერთნაირი სახელები შეგვიძლია გამოვიყენოთ, რადგან პროგრამა და ფუნქცია სრულიად გამიჯნულნი არიან ურთიერთისგან. თუმცა პროგრამისათვის ხელმისაწვდომია მხოლოდ ფუნქციის გამოსავალი არგუმენტის მნიშვნელობა.
5. თუ ფუნქცია შედეგად იძლევა ერთზე მეტი სიდიდის მნიშვნელობას, ეს სიდიდეები უნდა იყოს მითითებული როგორც ვექტორი ბრძანებაში function ქვემოთმოყვანილ მაგალითად:
   function [dist, vel, accel] = motion (x)
   ფუნქცია უნდა ითვლიდეს სამივე სიდიდეს.

6. თუ ფუნქციას აქვს რამდენიმე შესავალი არგუმენტი, ისინი ჩამოთვლილი უნდა იყოს function ბრძანებაში: **function** error = mse(w,d)
7. სპეციალური ცვლადები nargin და nargout უნდა გამოვიყენოთ, რომ განვსაზღვროთ მიცემული ფუნქციის შესავალი და გამოსავალი არგუმენტების რაოდენობა. ეს ცვლადები მისაწვდომია მხოლოდ ფუნქციის შიგნით.

გავიხსენოთ, რომ ბრძანება **what** ჩამოთვლის ყველა M-ფაილს მოცემულ სამუშაო სივრცეში. ბრძანება **type,** რომელსაც მოყვება ფაილის სახელი გამოგვიყვანს ამ ფაილის შინაარსს ეკრანზე. თუ გაფართოება .m არ მივუთითეთ, **type** ბრძანება ავტომატურად გულისხმობს, რომ ეს .m ფაილია.

პრობლემა: ჰიდროაკუსტიკური სიგნალი

ჰიდროაკუსტიკური კვლევა საჭიროებს ბგერითი სიგნალის შექმნას, მის გავრცელებას წყალში და არეკლილი სიგნალის მიღებას. ოკეანოლოგიური ამოცანები მოიცავს ოკეანის ტოპოგრაფიის, რუქის აგების და ბიოლოგიური სიგნალების გაზომვის პრობლემებს. ინდუსტრიული ჰიდროაკუსტიკა იკვლევს თევზების ადგილსამყოფელს, ნავთობისა და მინერალების გავრცელების არეალს. საზღვაო ჰიდროაკუსტიკა გამოიყენება წყალქვეშა გემების ადგილმდებარეობის განსაზღვრისათვის. აქტიური ჰიდროაკუსტიკური სისტემა აწარმოებს სიგნალის გადაცემას, რომელიც ჩვეულებრივ გარკვეული სიხშირის სინუსოიდაა. ამ სიგნალის ანარეკლი ან ექო უკან ბრუნდება და ხდება მისი მიღება და ანალიზი. პასიური ჰიდროაკუსტიკური სისტემა სიგნალებს არ გადასცემს, მაგრამ აგროვებს და აანალიზებს მათ.

პირველ რიგში განვიხილავთ სინუსოიდას, რადგან იგი ძირითადი სიგნალია, რომელიც ჰიდროაკუსტკურ სისტემაში გამოიყენება. შემდეგ შევადგენთ MATLAB პროგრამას ჰიდროაკუსტიკური სიგნალის მოდელირებისათვის.

სინუსოიდა

ჩვენ ვიცით **sin** – კუთხის ფუნქცია. სინუსოიდა არის **sin** ფუნქცია , რომელიც ჩაწერილია როგორც დროის ფუნქცია:

$$g(t) = \sin(2\pi ft)$$

სადაც f სინუსოიდის სიხშირეა. მისი ერთეულია ციკლი/წამში, ან ჰერცი. (შევნიშნავთ, რომ 2πft-ის ერთეულია რადიანი.
თუ სინუსოიდის სიხშირე 5 ჰერცია, გვაქვს

$$g(t) = \sin(2\pi 5t) = \sin(10\pi t)$$

თუ ავაგებთ ამ ფუნქციის გრაფიკს, ვნახავთ, წამში მართლაც ხუთი რხევაა. ნახ 3.7. სინუსოიდის პერიოდი ეწოდება დროის შუალედს, რომელშიც იგი ერთ სრულ რხევას ასრულებს. ასე, რომ ამ სინუსოიდის პერიოდია 0.2 წმ. სიხშირესა და პერიოდს შორის ასეთი დამოკიდებულება არსებობს:

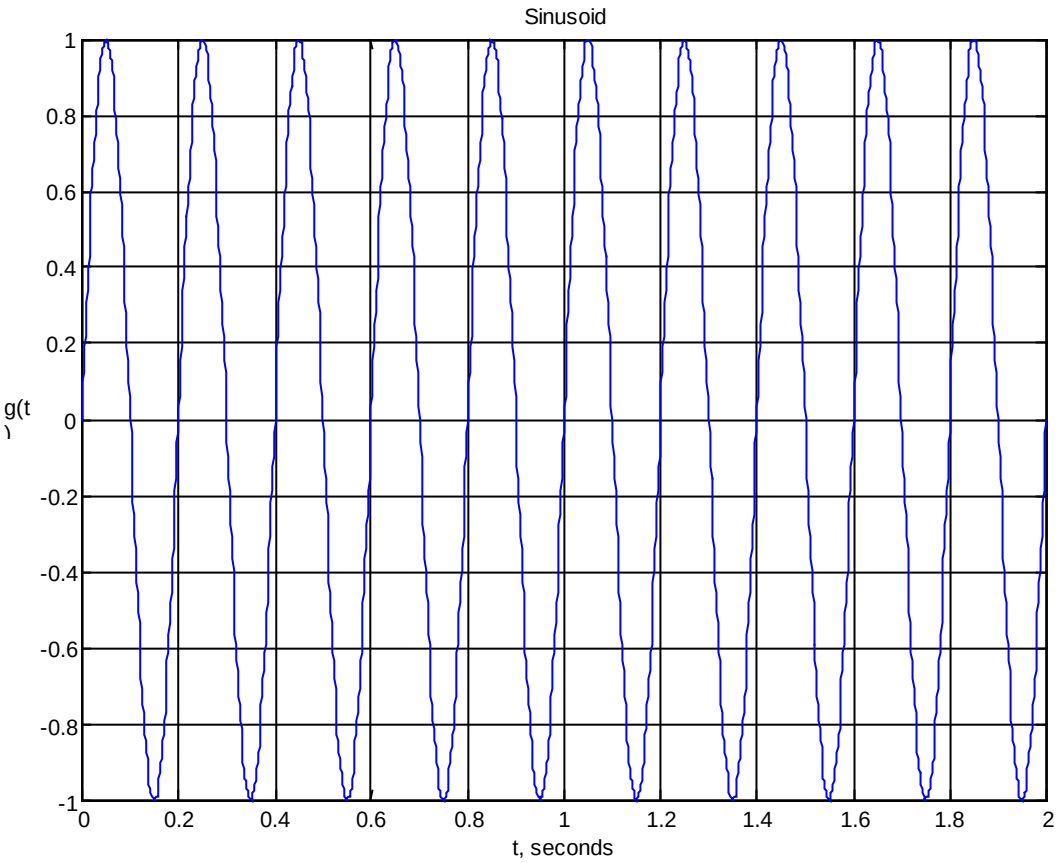
$$f = \frac{1}{p}; \qquad p = \frac{1}{f};$$

სადაც f სიხშირეა ჰერცებში, P პერიოდი წამებში.
თუ სინუსოიდას გავამრავლებთ სკალარზე A, განტოლება ჩაიწერება ფორმით:

$$g(t) = A\sin(2\pi ft)$$

ამ სკალარს უწოდებენ სინუსოიდის ამპლიტუდას. სინუსოიდა ფაზური წანაცვლებით ϕ ასე ჩაიწერება:

$$g(t) = A\sin(2\pi ft + \phi)$$



ნახ. 3.7 5 ჰერციანი სინუსოიდა

თუ ფაზური წანაცვლება $\pi/2$ რადიანის ტოლია, გვექნება:

$$A \cdot \sin\left(2\pi ft + \frac{\pi}{2}\right) = A \cdot \cos(2\pi ft)$$

სინუსოიდა არის დროის ფუნქცია, რომელიც შეიძლება ჩაიწეროს სინუსის ან კოსინუსის საშუალებით და შეიძლება ჰქონდეს ან არ ჰქონდეს ფაზური წანაცვლება.

ჰიდროაკუსტიკური სიგნალის გენერირება

აქტიური ჰიდროაკუსტიკური სისტემა გადასცემს განსაზღვრული სიხშირის სიგნალს. ამ სიგნალის ანარეკლი ან ექო უკანვე მიიღება და ხდება მისი ანალიზი გარკვეული ამოცანების შესასწავლად. პასიური სისტემა მხოლოდ არეკლილი სიგნალის მიღებას და ანალიზს აწარმოებს. ჰიდროაკუსტიკურ სისტემაში ძირითადად სინუსოიდა გამოიყენება. სინუსოიდა ასეთი სახითაა მოცემული

$$s(t) = \begin{cases} \sqrt{\frac{2E}{PD}} \cos(2\pi f_c t) & 0 \le t \le PD \\ 0 & elsewhere \end{cases}$$

სადაც E გადაცემის ენერგიაა, PD – პულსის ხანგრძლივობა წამებში, f_c – სიხშირე ჰერცებში.
პულსის ხანგრძლივობა შესაძლოა იყოს მილიწამის ნაწილიდან რამდენიმე წამამდე. სიხშირული დიაპაზონი რამდენიმე ასეული ჰერციდან ათეულობით კილოჰერცამდეა.

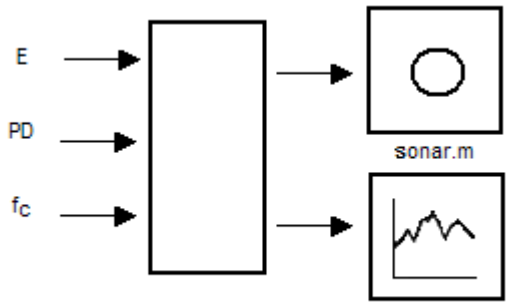
დაწერეთ MATLAB პროგრამა, რომელიც საშუალებას მოგცემთ მიაწოდოთ პროგრამას E, PD, f_c მნიშვნელობები ჰიდროაკუსტიკური სიგნალის გენერირებისათვის, შეინახავს სიგნალის მნიშვნელობებს ფაილში sonar.mat. სიგნალის ათვლის ხანგრძლივობა უნდა ფარავდეს პულსის ხანგრძლივობას, ყოველი პერიოდის განმავლობაში 10 ანათვალი უნდა იქნას აღებული. გარდა ამისა, სიგნალს უნდა დაემატოს 200 წერტილი სიჩუმისა პულსის შეწყვეტის შემდეგ.

1. ამოცანის დასმა

დაწერეთ პროგრამა იმისათვის, რომ შეიქმნას გარკვეული ხანგრძლივობის ჰიდროაკუსტიკური სიგნალი, რომელიც შეიცავს 10 ანათვალს ყოველი პერიოდის განმავლობაში.

2. INPUT/OUTPUT აღწერა

სიდიდეები: E გადაცემის ენერგია ჯოულებში, PD – პულსის ხანგრძლივობა წამებში, f_c სიხშირე ჰერცებში არის INPUT, შესავალი მნიშვნელობები. გამოსავალი მნიშვნელობაა მონაცემთა ფაილი სახელით sonar.mat, რომელიც შეიცავს დრიოს და სიგნალის შესაბამის მნიშვნელობებს. ავაგებთ აგრეთვე სიგნალის გრაფიკს:



ნახ. 3.8 I/ O დიაგრამა

3. სახელდახელო ამოხსნა

ვისარგებლოთ შემდეგი მონაცემებით:

E = 500 ჯოული
PD = 5 მილიწამი
f_c = 3.5 კილოჰერცი

სინუსოიდის პერიოდია 1/3500 = 0.3 მილიწამი. იმისათვის, რომ გვქონდეს 10 ანათვალი პერიოდში, ათვლის ინტერვალი უნდა იყოს 0.03 მილიწამი. პულსის ხანგრძლივობაა 5 მილიწამი და ამიტომ გვჭირდება 167 (5/0.03) ანათვალი შემდეგი სიგნალისა:

$$s(t) = \sqrt{\frac{2E}{PD}}\cos(2\pi f_c t) = \sqrt{\frac{1000}{0.005}}\cos(2\pi(3500)t)$$

ქვემოთ მოყვანილია სიგნალის პირველი რამდენიმე მნიშვნელობა:

t(მწმ)	s(t)
0.00	447.2
0.03	353.4
0.06	11.2
0.09	-177.6
0.12	-391.9
0.15	-441.7
0.18	-306.1
0.21	-42.1
0.24	239.6
0.27	420.8
0.30	425.3
0.33	251.4

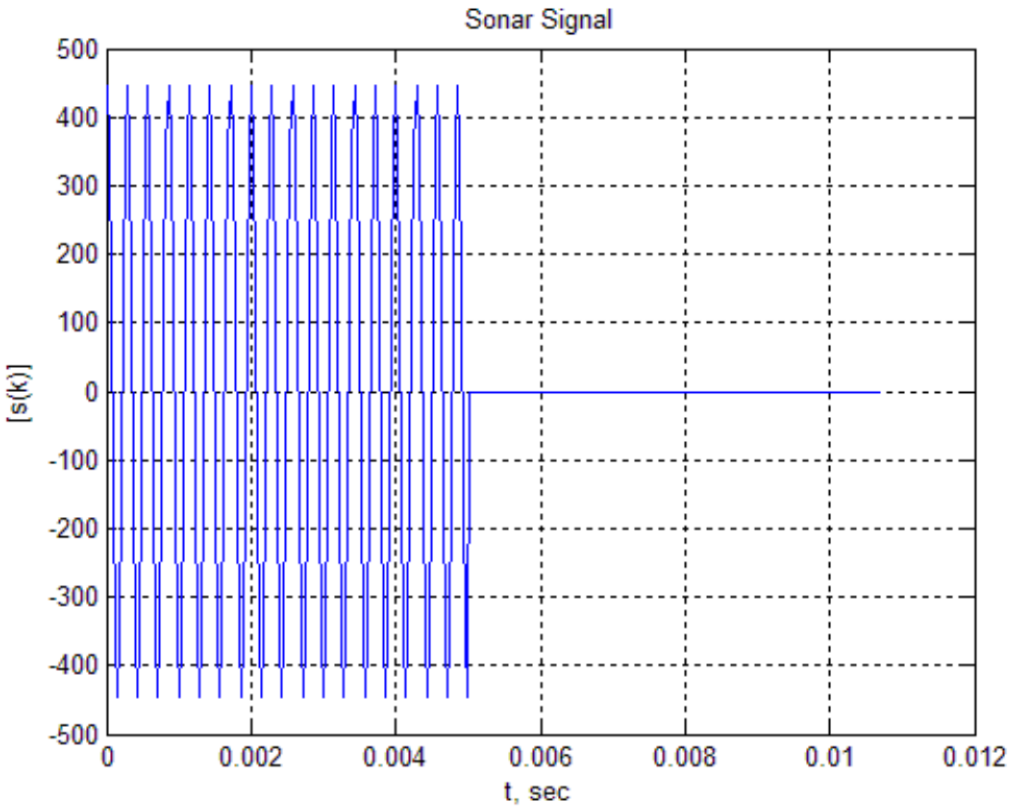
ამას დავუმატებთ 200 წერტილს, როცა სიგნალის მნიშვნელობა 0 –ის ტოლია.

4. MATLAB ამოხსნა

ამ პროგრამის ყველაზე მნიშვნელოვანი ნაწილია სიჩუმისათვის დროის შესაბამისი მნიშვნელობების გამოთვლა. დროის მნიშვნელობებს უნდა დავუმატოთ 200 წერტილი. ჯერ შევქმნით სკალარს სახელად silence_start, რომელიც მიიღებს დროის იმ მნიშვნელობას, როცა სიგნალი შეწყდა და დაიწყო სიჩუმე. შემდეგ შევქმნით სიჩუმის პერიოდის საბოლოო მნიშვნელობას და მათ შორის ორწერტილოვანი ოპერატორით შევქმნით ვექტორს, რომელიც შეიცავს ამ ინტერვალში დროის მნიშვნელობებს. ამის შემდეგ შევქმნით ახალ ვექტორს სიგნალის მნიშვნელობებისათვის, რომელიც შეიცავს ზუსტად იმდენ ნულოვანი მნიშვნელობის ელემენტს, რამდენსაც დროის ახალი ვექტორი.

```
%               This program generates a sonar signal
%               using values obtained from the user for
%               energy, pulse duration and frequency.
%               An additional 200 points of silence are added
%               to the signal
%               before it is stored.
  energy = input('Enter energy in joules ');
  duration =input('Enter puls duration in seconds ');
  fc =input('Enter frequency in Hz ');
%
%Generate sonar signal
%
  A=sqrt(2*energy/duration);
  period=1/fc;
  t_incr=period/10;
  t=[0:t_incr:duration];
  s=A*cos(2*pi*fc*t);
%
```

```
%generate and add 200 points of silence
%
last=length(t);
silence_start=t(last)+t_incr;
silence_end=t(last)+200*t_incr;
silence=[silence_start:t_incr:silence_end];
t=[t silence];
s=[s zeros(size(silence))];
%
%Save and plot sonar signal
%
save sonar t s
plot(t,s),...
    title('Sonar Signal'),...
    xlabel('t, sec'),...
    ylabel('[s(k)]'),...
    grid
```



ნახ. 3.9 ჰიდროაკუსტიკური სიგნალი

3.5 კომპლექსური რიცხვები

მრავალი საინჟინრო ამოცანა მოითხოვს ვიპოვოთ შემდეგი განტოლების ფესვები:

y = f(x)

სადაც ფესვები არის x –ის ის მნიშვნელობები, რომელთათვისაც y = 0.

კვადრატული განტოლებისათვის (მეორე რიგის პოლინომი) ვიყენებთ სპეციალურ ფორმულას. განვიხილოთ კვადრატული განტოლება:

$$f(x) = x^2 + 3x + 2$$

თუ დავშლით მამრავლებად, მივიღებთ:

$$f(x) = (x+1)\cdot(x+2)$$

როგორც ვნახეთ ამ განტოლების ფესვებია -1 და -2.
ახლა განვიხილოთ კვადრატული განტოლება:

$$f(x) = x^2 + 3x + 3$$

მისი ფესვებია:

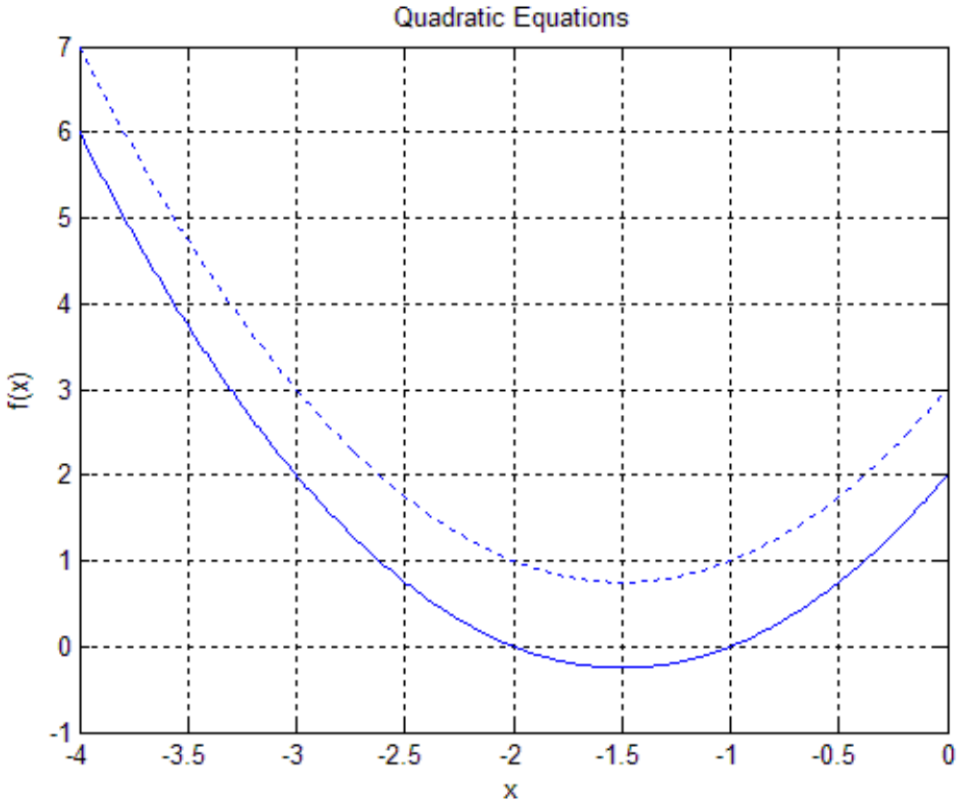
$$x_1 = -1.5 + 0.87\sqrt{-1}$$
$$x_2 = -1.5 - 0.87\sqrt{-1}$$

იმისათვის, რომ ამ გამოსახულებას აზრი ჰქონდეს, უნდა განვსაზღვროთ უარყოფითი რიცხვიდან კვადრატული ფესვის მნიშვნელობა. ამისათვის არსებობს კომპლექსურ რიცხვთა სიმრავლე, $a + ib$, სადაც $I = \sqrt{-1}$. თუ (b =0 შემთხვევაში იგი ნამდვილი რიცხვია).
ნახ 3.10 წარმოადგენს ორივე კვადრატული პოლინომის გრაფიკს. როგორც მოსალოდნელი იყო, პირველი მათგანის გრაფიკი კვეთს x ღერძს ორ წერტილში, რომლებიც მის რეალურ ფესვებს შეესაბამება, ხოლო მეორე პოლინომის გრაფიკი არ გადაკვეთს x ღერძს, რადგან მას გააჩნია ორი კომპლექსური ფესვი.

განვიხილოთ n რიგის პოლინომის ზოგადი სახე:

$$a_1x^n + a_2x^{n-1} + a_3x^{n-2} + \cdots + a_{n-2}x^2 + a_{n-1}x + a_n = 0$$

n რიგის პოლინომს აქვს n ფესვი, ზოგიერთი მათგანი ნამდვილია, ზოგიც კომპლექსური. ამ წიგნის მე-10 თავში განვიხილავთ MATLAB ფუნქციას პოლინომის ფესვების საპოვნელად. ამ თავში კი განვიხილავთ მოქმედებებს კომპლექსურ რიცხვებზე და MATLAB ფუნქციებს, რომლებიც კომპლექსურ რიცხვებს უკავშირდება.



ნახ. 3.10 ორი სხვადასხვა კვადრატული ფუნქციის გრაფიკი

3.1.7 არითმეტიკული ოპერაციები კომპლექსურ რიცხვებზე

მრავალი საინჟინრო პრობლემის ამოხსნისას კომპლექსური რიცხვები მნიშვნელოვან როლს ასრულებს:

მოქმედება	შედეგი
$c_1 + c_2$	$(a_1+a_2)+i(b_1+b_2)$
$c_1 - c_2$	$(a_1-a_2)+i(b_1-b_2)$
$c_1 * c_2$	$(a_1 a_2 - b_1b_2)+i(a_1b_2+ a_2b_1)$
c_1 / c_2	$((a_1 a_2 + b_1b_2)/(a_2^2+b_2^2)) +$ $i((a_2b_1 + b_2 a_1)/(a_2^2+b_2^2))$
$\|c_1\|$	$\sqrt{a_1^2+b_1^2}$ (magnitude)
c_1^*	$a_1 - ib_1$ (conjugate of c_1)

სადაც $c_1 = a_1 + ib_1$ $c_2 = a_2 + ib_2$

ბრძანება x = 1 – 0.5*i გვაძლევს კომპლექსურ რიცხვს x. როცა ვაწარმოებთ მოქმედებებს კომპლექსურ რიცხვებზე, MATLAB ავტომატურად ასრულებს ზემოთმოყვანილ ცხრილში აღწერილ მოქმედებებს. თუ მოქმედება სრულდება ნამდვილსა და კომპლესურ რიცხვს შორის, MATLAB გულისხმობს, რომ ნამდვილი რიცხვი ეს არის კომპლექსური რიცსვი, რომლის წარმოსახვითი ნაწილი 0 –ის ტოლია.

MATLAB – ს აქვს რამდენიმე ფუნქცია კომპლექსური რიცხვებისათვის:

real(x)	ითვლის კომპლექსური რიცხვის ნამდვილ ნაწილს

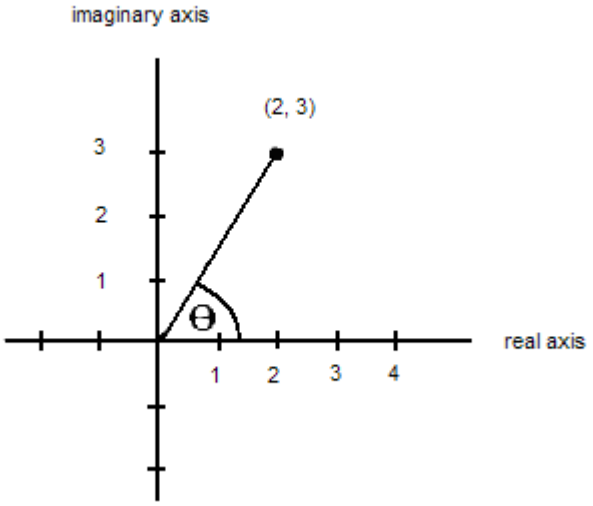
imag(x)	ითვლის კომპლექსური რიცხვის არმოსახვით ნაწილს
conj(x)	ითვლის კომპლექსური x რიცხვის შეუღლებულს
abs(x)	ითვლის კომპლექსური რიცხვის მოდულს.
angle(x)	ითვლის კუთხეს atan2(imag(x), real(x)) ფუნქციის საშუალებით

ეს ფუნქციები აადვილებს კომპლექსური რიცხვის ერთი ფორმიდან მეორეში გადაყვანას.

3.1.8 კოორდინატთა დეკარტის და პოლარული სისტემა

კომპლექსურ რიცხვთა სისტემა შეგვიძლია წარმოვადგინოთ როგორც სიბრტყე ნამდვილი და წარმოსახვითი ღერძებით. რეალური რიცხვები წარმოადგენს x ღესძს, წარმოსახვითი რიცხვები (ნამდვილი ნაწილის გარეშე) – y ღერძს, ხოლო ყველა სხვა კომპლექსური რიცხვი, რომელსაც გააჩნია როგორც ნამდვილი, ისე წარმოსახვითი ნაწილი, დაიკავებს შესაბამის ადგილს სიბრტყეზე. როცა ვიხილავთ კომპლექსურ რიცხვს ფორმით 2 – i3, საქმე გვაქვს კომპლექსური რიცხვის მართკუთხა აღნიშვნასთან. კომპლექსური რიცხვები შეგვიძლია წარმოვადგინოთ θ კუთხისა და r რადიუსის საშუალებით – პოლარული აღნიშვნა. ნახ. 3.11 –დან მარტივად შეგვიძლია განვსაზღვროთ დამოკიდებულება მართკუთხა და პოლარულ კოორდინატებს შორის:

$$r = \sqrt{a^2 + b^2}; \theta = \tan^{-1}\frac{b}{a}$$



ნახ. 3.11 კონპლექსური სიბრტყე

პოლარულიდან დეკარტის სისტემაში გადასაყვანი ფორმულები:

$$a = r \cos \theta$$
$$b = r \sin \theta$$

კალკულატორს შეუძლია შეარულოს გადასაყვანი ფორმულების გამოთვლა. აქ საჭიროა სიფრთხილე, რომ ზუსტად განვსაზღვროთ რომელ მეოთხედში უნდა მდებარეობდეს θ კუთხე. ხშირ შემთხვევაში კალკულატორი ავტომატურად იძლება კუთხის მნიშვნელობას პირველ მეოთხედში, მისი ყოველგვარი ანალიზის გარეშე. MATLAB-ში ფუნქცია **atan(x)** კუთხეს გვაძლევს შუალედში -π/2 - +π/2, მაშინ როცა **atan2(y,x)** გვაძლევს y/x ტანგენსის შებრუნებულს -π - +π შუალედში. ფუნქცია **angle(x)** არგუმენტია კომპლექსური რიცხვი x და გვაძლევს სიდიდეს atan2(imag(x), real(x)).

თუ x კომპლექსური რიცხვია მისი მოდული და ფაზა გამოითვლება შემდეგი ბრძანებით:

```
r = abs(x);
theta = angle(x);
```

თუ გვსურს გამოვითვალოთ კომპლექსური რიცხვი, როცა მოცემულია მისი მოდული და ფაზა:

```
y = r*exp(I*theta);
```

კომპლექსური რიცხვის ნამდვილი და წარმოსახვითი ნაწილი შეგვიძლია გამოვითვალოთ შემდეგი ბრძანებებით:

```
a = real(x);
b = imag(x);
```

რომ წარმოვადგინოთ კომპლექსური რიცხვი მისი ნამდვილი და წარმოსახვითი ნაწილის საშუალებით, დავწერთ: y = a + i*b.

სავარჯიშო

წარმოადგინეთ კომპლექსური რიცხვები პოლარული ფორმით, შეამოწმეთ პასუხები MATLAB საშუალებით:

1. 3 – i2
2. –I
3. –2
4. 0.5 + i

გარდაქმენით კომპლექსური რიცხვები ექსპონენციალურიდან მართკუთხა ფორმაში. შეამოწმეთ პასუხები MATLAB საშუალებით. hh

5. ei
6. eiπ0.75
7. 0.5ei2.3
8. 3.5 ei3π

3.1.9 ეილერის ფორმულა

კომპლექსური რიცხვების ზოგიერთი მნიშვნელოვანი თვისების მისაღებად განვიხილოთ Maclaurin მწკრივი:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} - \cdots$$

ვთქვათ x არის წარმოსახვითი სიდიდე ib, მაშინ გვექნება:

$$e^{ib} = 1 + ib + \frac{(ib)^2}{2!} + \frac{(ib)^3}{3!} - \cdots = 1 + ib - \frac{b^2}{2!} - i\frac{b^3}{3!} + \frac{b^4}{4!} - \frac{b^5}{5!} + \cdots$$

საბოლოოდ მივიღებთ:

$$e^{ib} = \cos b + i \sin b$$

ეს ძალიან მნიშვნელოვანი ფორმულა და ეილერის ფორმულას უწოდებენ. მისგან მიიღება კიდევ 2 გამოსახულება:

$$\sin\theta = \frac{e^{i\theta} - e^{-i\theta}}{2i}$$

$$\sin\theta = \frac{e^{i\theta} + e^{-i\theta}}{2}$$

ეილერის ფორმულის საშუალებით კომპლექსური რიცხვი შეგვიძლია გამოვსახოთ პოლარულ და მართკუთხა ფორმით.

$$a + ib = (r\cos\theta) + i(r\sin\theta) = r(\cos\theta \_ i\sin\theta)$$

ეილერის ფორმულის გამოყენებით მივიღებთ:

$$a + ib = re^{i\theta}$$

sadac $r = \sqrt{a^2 + b^2}; \theta = \tan^{-1}\frac{b}{a}; a = r\cos\theta; b = r\sin\theta$

ამ თავში განვიხილეთ მათემატიკური ოპერაციები, MATLAB ფუნქციები მატრიცების შესაქმნელად და უკვე არსებული მატრიცებისაგან ახალ მატრიცების საწარმოებლად. განვიხილეთ მათემატიკური ოპერაციები მატრიცების შესაბამის ელემენტებს შორის ე.წ. მასივური ოპერაციები. ვისწავლეთ აგრეთვე როგორ შევქმნათ ახალი ფუნქცია MATLAB-ში m-ფაილის სახით. განხილული იქნა აგრეთვე კომპლექსური რიცხვები და მათზე წარმოებული მათემატიკური ოპერაციები. განვიხილეთ რამდენიმე ამოცანა, რომელიც

დაკავშირებული იყო საკომუნიკაციო სიგნალის ექოსთან და ჰიდროაკუსტიკური სიგნალის მოდელირებასთან.

ყველა სპეცსიმბოლო, ბრძანება და ფუნქცია, რომელიც ამ თავში იქნა განხილული მოკლე მიმოხილვითირთ.

სპეცსიმბოლოები

+	სკალარული და მასივური შეკრება
-	სკალარული და მასივური გამოკლება
*	სკალარული გამრავლება
.*	მასივური გამრავლება
/	სკალარული მარჯვნივ გაყოფა
./	მასივური მარჯვნივ გაყოფა
\	სკალარული მარცხნივ გაყოფა
.\	მასივური მარცხნივ გაყოფა
^	სკალარული ახარისხება
.^	მასივური ახარისხება

ბრძანებები და ფუნქციები:

/	სკალარული მარჯვნივ გაყოფა
./	მასივური მარჯვნივ გაყოფა
abs	გამოითვლის მოდულს
acos	ითვლის არკკოსინუსს
acosh	ითვლის ჰიპერბოლურ კოსინუსს
angle	ითვლის კომპლექსური რიცხვის ფაზურ კუთხეს
ans	აგებს გამოსახულების მნიშვნელობას
asin	ითვლის არკსინუსს
asinh	ითვლის ჰიპერბოლური სინუსის შებრუნებულს
atan	ითვლის არკტანგენსს მეოთხედში
atan2	ითვლის არკტანგენსს მეოთხედში
atanh	გამოითვლის ჰიპერბოლური ტანგენსის შებრუნებულს
ceil	ამრგვალებს ∞ მიმართულებით
clock	გვაძლევს მიმდინარე დროს
conj	გვაძლევს კომპლექსური რიცხვის შეუღლებულს
cos	გვაძლევს კუთხის კოსინუსს
cosh	გვაძლევს ჰიპერბოლურ კოსინუს
date	გვაძლევს მიმდინარე თარიღს
eps	ათწილადურ სიზუსტეს
exp	ითვლის ხარისხს e ფუძით
eye	ქმნის ერთეულოვან მატრიცას
function	ქმნის ახალ ფუნქციას MATLAB-ში
fix	ამრგვალებს ნულის მიმართულებით

floor	ამრგვალებს - ∞ მიმართულებით
i	კვადრატული ფესვი –1 დან
imag	ითვლის კომპლექსური რიცხვის წარმოსახვით ნაწილს
inf	უსასრულობა ∞
j	კვადრატული ფესვი –1 –დან
log	გამოითვლის ლოგარიტჰმს ნატურალური ფუძით
log10	გამოითვლის ლოგარიტჰმს 10 ფუძით
magic	აწარმოებს მაგიურ კვადრატს
NaN	განუზღვრელობა
nargin	გვაძლევს ფუნქციის არგუმენტების რაოდენობას
nargout	განსაზღვრავს ფუნქციის გამოსავალი არგუმენტების რაოდენობას
ones	ქმნის მატრიცას, რომლის ყველა ელემენტი 1-ის ტოლია
pascal	ქმნის პასკალის სამკუთხედა მატრიცაში
pi	π
real	ითვლის კომპლექსური რიცხვის ნამდვილ ნაწილს
rem	ითვლის მთელი რიცხვების გაყოფის შედეგად მიღებულ ნაშთს
round	ამრგვალებს უახლოეს მთელამდე
sign	გვაზლევს –1, 0 ან 1
sin	ითვლის კუთხის სინუსს
sinh	ითვლის ჰიპერბოლურ სინუსს
sqrt	ითვლის კვადრატუ ფესვს
tan	ითვლის კუთხის ტანგენსს
tanh	ითვლის ჰიპერბოლურ ტანგენსს
zeros	ქმნის მატრიცას, რომლის ყველა ელემენტი 0-ის ტოლია

ამოცანები

პრობლემა 1-8 უკავშირდება ამ თავში განხილულ ამოცანებს, ხოლო 9-23 მიეკუთვნება განსხვავებულ საინჟინრო პრობლემებს.

ეს პრობლემები დაკავშირებულია ექოს შემცველ სიგნალთან. გარდა იმისა, რომ შევქმნათ სიგნალი, რომელიც ექოს შეიცავს, უნდა შევძლოთ ექოს შემცველი სიგნალის მოდიფიცირება.

1. აწარმიეთ და ააგეთ 500 წერტილი სიგნალისა, რომელიც შეიცავს საწყის სიგნალს, რომელიც წარმოდგენილია ამ თავში განხილულ პრობლემაში ექო, ექოს, რომელიც შესუსტებულია ფაქტორით 0.3 და დაგვიანებულია 3 წამით.
2. შექმენით და ააგეთ 500 წერტილი სიგნალისა, რომელიც შეიცავს წინა ამოცანაში მითითებულ საწყის სიგნალს, ჩახვეულ ექოს სკალირებულს ფაქტორით 0.6 და დაგვიანებულს 5.5 წამით
3. აწარმიეთ და ააგეთ 500 წერტილი სიგნალისა, რომელიც შეიცავს საწყის სიგნალს, რომელიც წარმოდგენილია ამ თავში განხილულ პრობლემაში ექო და კიდევ 2 ექოს. პირველი დაგვიანებულია 0.5 წამით და შესუსტებულია ფაქტორით 0.7, მეორე – დაგვიანებულია 4 წამით და შესუსტებული ფაქტორით 0.1.

4. შექმენით და ააგეთ 500 წერტილი სიგნალისა, რომელიც შეიცავს მხოლოდ3 პრობლემაში ასახულ 2 ექოს.
5. აიღეთ სიგნალი ექოსთან ერთად, რომელიც განხილულია ამ თავში, ააგეთ ახალი სიგნალი, რომელიც შეიცავს სიგნალის და ექოების აბსოლუტურ სიდიდეებს.
6. აიღეთ სიგნალი ექოსთან ერთად, რომელიც განხილულია ამ თავში, ააგეთ ახალი სიგნალი, რომელიც შეიცავს სიგნალის და ექოების კვადრატში აყვანილ მნიშვნელობებს.

ჰიდროაკუსტიკური სიგნალი.

7. შექმენით და ააგეთ ჰიდროაკუსტიკური სიგნალი, რომელიც შეიცავს სიგნალს განხილულს ამ თავის შესაბამის განყოფილებაში (ჰიდროაკუსტიკური სიგნალი), რომელსაც მოჰყვება სიჩუმის 200 წერტილი და შემდეგ იგივე სიგნალი 2 მილიწამის ხანგრძლივობით.
8. შექმენით და ააგეთ ჰიდროაკუსტიკური სიგნალი, რომელიც შეიცავს ამ თავის შესაბამის განყოფილებაში განხილულ სიგნალს (ჰიდროაკუსტიკური სიგნალი), რომელსაც მოჰყვება სიჩუმე 1 მილიწამის ხანგრძლივობით

ფორმულები გვიჩვენებს გრადუსებში გამოსახულ ტემპეტატურათა დამოკიდებულებას სხვადასხვა სისტემაში: ფარენჰეიტის გრადუსი(T_F), ცელსიუსის გრადუსი(T_C), კელვინის გრადუსი(T_K) და რანკინის გრადუსი(T_R):

$$T_F = T_R - 459.67^oR$$
$$T_F = 9/5\ T_C + 32^oF$$
$$T_R = 9/5\ T_K$$

9. დაწერეთ პროგრამა იმისათვის, რომ შეიქმნას ცხრილი ფარენჰეიტის ტემპერატურიდან ცელსიუსის ტემპერატურაში გადასაყვანად 0^o დან 100^o -მდე ყოველი 5 გრადუსის შემდეგ.

10. დაწერეთ პროგრამა იმისათვის, რომ შეიქმნას ცხრილი ფარენჰეიტის ტემპერატურიდან კელვინის ტემპერატურაში გადასაყვანად 0^o დან 100^o -მდე ყოველი 5 გრადუსის შემდეგ.

ბაქტერიების გამრავლება. ბაქტერიების რაოდენობრივი ზრდა შეიძლება მოდელირებული იქნას შემდეგი ფორმულის მიხედვით:

$$Y_{new} = Y_{old}\ e^{1.386\ t}$$

სადაც Y_{new} ბაქტერიების ახალი რაოდენობა მოცემულ გარემოში, Y_{old} ბაქტერიების საწყისი რაოდენობაა, t დრო საათებში.

11. ისარგებლეთ ამ განტოლებით რომ იწინასწარმეტყველოთ ბაქტერიების რაოდენობა 6 საათის შემდეგ, თუ მათი საწყისი რაოდენობა 1-ის ტოლია. დაბეჭდეთ ცხრილი, რომელიც უჩვენებს ბაქტერიების რაოდენობას ყოველი საათის შემდეგ.
12. შეცვალე 11 დავალება ისე, რომ მომხმარებელმა თვითონ შეძლოს დროის მითითება საათებში.
13. შეცვალე 11 დავალება ისე, რომ მომხმარებელმა მიუთითოს დრო წუთებში, თუმცა ფორმულა დროს სათებში აღიქვამს.

14. შეცვაე 11 დავალება ისე, რომ მომხმარებელს შეეძლოს ბაქტერიების საწყისი რაოდენობის მითითება.
15. შეცვალე 11 დავალება ისე, რომ მომხმარებელმა მიუთითოს დროის ორი მნიშვნელობა, პროგრამამ კი დაითვალოს დროის ამ ინტერვალში წარმოქმნილი ბაქტერიების რაოდენობა.

დათარიღება ნახშირბადის შემცველობის მიხედვით.

ეს არის მეთოდი, რომლის საშუალებითაც ხდება ორგანული ნაერთების ნაშთის, მაგალითად ნიჟარა, მცენარის თესლი, ასაკის დადგენა. ეს მეთოდი ადგენს რადიაციული ნახშირბადის carbon14 წილობრივ შემცველობას მოცემულ ნიმუშში. განტოლება იძლევა ნარჩენის ასაკს წლებში:

$$age = \frac{-\log_e(carbon14 \cdot proportion \cdot remaining)}{0.0001216}$$

carbon14 სიცოცხლის ხანგრძლივობა დაახლოებით 11400 წელია. თუ პროპორციულ შემცველობას ავიღებთ 0.5, ასაკი ტოლი იქნება 5700.22.

16. დაწერეთ პროგრამა, რომელიც მომხმარებელს საშუალებას მისცემს მიუთითოს ნარჩენი carbon14 პროპორციული შემცველობა ეს სიდიდე უნდა იყოს 0 და 1 –ს შორის). დაბეჭდეთ ნიმუშის ასაკი.
17. შეცვალე 16 დავალება ისე, რომ მიღებული შედეგი დამრგვალდეს უახლოეს წლამდე.
18. შეცვალე 16 დავალება ისე, რომ მიღებული შედეგი დამრგვალდეს უახლოეს საუკუნემდე.

სიხშირული გაზომვები.

სინუსოიდის სიხშირე შეიძლება მიცემული იყოს ციკლების რაოდენობით წამში (ჰერცი) ან რადიანებით წამში, სადაც ერთი ციკლი წამში ტოლია 2π რადიანისა წამში.

19. დაწერეთ პროგრამა, რომელიც საშუალებას მოგცემთ მიუთითოთ სიხშირე ციკლი/წამში და გადაიყვანოთ იგი ერთეულში რადიანი/წამში.
20. დაწერეთ პროგრამა, რომელიც საშუალებას მოგცემთ მიუთითოთ სიხშირე რადიანი/წამში და გადაიყვანოთ იგი ჰერცებში.
21. ნორმირებული სიხშირე ტოლია ωT, სადაც ω არის სიხშირე რადიანი/წამში და T გამოსახულია წამებში. ე.ი. ნორმირებული სიხშირის ერთეულია რადიანი. დაწერეთ პროგრამა, რომელიც საშუალებას იძლევა მიუთითოთ სიხშირე რადიანი/წამში და T სიდიდე და გამოითვალოთ შესაბამისი ნორმირებული სიხშირე.
22. შეცვალეთ 21 დავალება ისე, რომ მიუთითოთ სიხშირე ჰერცებში.
23. შეცვალეთ 21 დავალება ისე, რომ ნორმირებული სიხშირე ყოველთვის მიიღოთ შუალედში 0 - 2π. (გამოაკელით მიღებულ შედეგს 2π ჯერადი მნიშვნელობა). მაგალითად: თუ ნორმირებული სიხშირე ტოლია 2.5π, ჩვენთვის სასურველი შედეგი უნდა იყოს 0.5π; თუ ნორმირებული სიხშირე ტოლია 7π, უნდა მივიღოთ - π. (მითითება: გაიხსენეთ ფუნქცია rem)

4 კონტროლის ოპერაციები

პრობლემა: ზეგამტარობა

მაგნიტურ ლევიტაციური ტრანსპორტი მოძრაობს, იმართება და ჩერდება ელექტრომაგნიტური ძალის მოქმედებით. ამ ტიპის ტრანსპორტს მიეკუთვნება სწრაფი მატარებელი (bullet train), რომელსაც შეუძლია განავითაროს სიჩქარე 300 მილი საათში. მატარებელი რელსებიდან დახლოებით 1 დუიმითაა დაცილებული განზიდულობის ძალის გამო, რომელსაც წარმოშობს მატარებელზე დამაგრებული ზეგამტარი მაგნიტი და კოჭა მიწაზე. ელექტრული სენსორები გრძნობენ მატარებლის მოახლოებას და არეგულირებენ ელექტრომაგნიტურ ველს. როცა სენსორებზე მატარებელი არ არის, ენერგია წყდება. ბორბლები შეიკეცება მატარებლის შიგნით, როცა ის მოძრაობს, მაგარმ ისინი დაეშვებიან, როცა მატარებელი უკლებს სიჩქარეს გასაჩერებლად.

საჭიროა ახალი მასალის გამოგონება, რომელიც იქნება უფრო მსუბუქი და უფრო მტკიცე ერთდროულად იმისათვის, რომ მოხდეს მომავლის სატრანსპორტო საშუალებათა სრულყოფა. ამ მიზნით შექმნილი ახალი მასალები გადიან მკაცრ და ხანგრძლივ ტესტირებას. ერთერთი მათგანია ტემპერატურის განაწილების ხასიათი. ამას დიდი მნიშვნელობა ენიჭება, როცა ეს მასალა გამოყენებული უნდა იქნას ძრავის ან მაღალ ტემპერატურული კომპონენტების მახლობლად.

შესავალი

MATLAB –ის პროგრამები, რომლებიც ჩვენ აქამდე დავწერეთ, შეიცავდა მხოლოდ მიმდევრობით საფეხურებს. ერთ ოპერაციას მეორე ცვლიდა ვიდრე არ დასრულდებოდა

საჭირო გამოთვლები. მაგრამ არსებობს მრავალი პრობლემა, სადაც საჭიროა შესარჩევი ბრძანება, რომელიც საშუალებას იძლევა გავუშვათ ბრძანებათა ერთი ციკლი, როცა გარკვეული პირობები სრულდება და ბრძანებათა სხვა ციკლი, როცა ეს პირობა არ სრულდება. ასევე დაგვჭირდება გავიმეოროთ ბრძანებათა გარკვეული ერთობლიობა რამდენიმეჯერ. ასეთი ტიპის ბრძანებებს კონტროლის ოპერაციებს უწოდებენ, რადგან ისინი საშუალებას გვაძლევს ვაკონტროლოთ თუ რომელი ბრძანებების გაშვებაა საჭირო.

if brZaneba

შერჩევის ბრძანება არის პირობითი მაკონტროლებელი ბრძანება, რომელიც საშუალებას იძლევა შემოწმდეს პირობა და ისე განისაზღვროს პროგრამის შემდგომი საფეხურები. შერჩევითი ბრძანების ძირითადი ფორმაა **if** ბრძანება. რამდენადაც იგი იყენებს შედარების და ლოგიკურ ოპერატორებს, განვიხილოთ როგორ ხორციელდება ისინი MATLAB–ში.

შედარების ოპერატორები

MATLAB აქვს შედარების 6 ოპერატორი:

შედარების ოპერატორი	ინტერპრეტაცია
<	ნაკლებია
<=	ნაკლებია ან ტოლია
>	მეტია
>=	მეტია ან ტოლია
==	ტოლია
~=	არ უდრის

მატრიცა და მატრიცული გამოსახულება შეიძლება შეგვხვდეს ამ ოპერატორთა ორივე მხარეს, მაგრამ შეიძლება შევადაროთ მხოლოდ ერთნაირი ზომის მატრიცები. შედეგად მიღებული იგივე ზომის მატრიცის შესაბამისი ელემენტი უდრის 1, თუ შერადების შედეგი ჭეშმარიტია და 0, თუ იგი მცდარია. გამოსახულებას, რომელიც შეიცავს შედარების ოპერატორს ლოგიკური გამოსახულება ეწოდება, რადგან შედეგად იძლევა მატრიცას, რომლის ელემენტების მნიშვნელობებია - 1 (ჭეშმარიტი) და 0 (მცდარი). შედეგად მიღებულ მატრიცას 0-1 მატრიცასაც უწოდებენ.

განვიხილოთ შემდეგი ლოგიკური გამოსახულება:

```
a < b
```

თუ a და b სკალარული სიდიდეებია, მაშინ შედეგად მივიღებთ 1, როცა იგი ჭეშმარიტია და 0, როცა იგი მცდარია. დავუშვათ a და b ვექტორებია:

```
a = [2 4 6]
b = [3 5 1]
```

ასეთ შემთხვევაში a < b ბრძანება მოგვცემს მატრიცას: [1 1 0], ხოლო a ~= b - [1 1 1].
ორი ლოგიკური გამოსახულება შეგვიძლია გავაერთიანოთ ლოგიკური ოპერატორით **და**, **ან**, **არ**. ეს ოპერატორები აღინიშნება სიმბოლოებით:

ლოგიკური ოპერატორი	სიმბოლო
და	&
ან	|
არ	~

ლოგიკური ოპერატორები საშუალებას იძლევა შევადაროთ ერთმანეთს შედარების ოპერატორებით მიღებული მატრიცები, მაგალითად:

```
a < b & b < c
```

ეს ოპერაცია შედეგს მოგვცემს, თუ a < b და b < c შედეგად მიღებული მატრიცები ერთნაირი ზომისაა. შესაბამისად მისი ელემენტები უდრის 1, როცა ორივე პირობა სრულდება, სხვა შემთხვევაში – 0.

როცა ორი ლოგიკური გამოსახულება შეერთებულია **ან (|)** ნიშნით, შედეგად მივიღებთ 0-1 მატრიცას, რომლის ელემენტები ტოლია 1, როცა ერთ-ერთი გამოსახულება მაინცაა ჭეშმარიტი, ხოლო 0-ის ტოლია, როცა ორივე გამოსახულება მცდარია. როცა ორი ლოგიკური გამოსახულება შეერთებულია ნიშნით - **და (&)**, შედეგად მიღებული მატრიცის შესაბამისი ელემენტი 1-ის ტოლია მხოლოდ და მხოლოდ მაშინ, როცა ორივე ლოგიკური გამოსახულება ჭეშმარიტია, სხვა შემთხვევაში 0-ის ტოლია. ცხრილი 4.1 შეიცავს A და B ლოგიკურ გამოსახულებებს შორის ლოგიკურ ოპერატორთა ყველა შესაძლო კომბინაციას:

A	B	~A	A | B	A & B
მცდარი	მცდარი	ჭეშმარიტი	მცდარი	მცდარი
მცდარი	ჭეშმარიტი	ჭეშმარიტი	ჭეშმარიტი	მცდარი
ჭეშმარიტი	მცდარი	მცდარი	ჭეშმარიტი	მცდარი
ჭეშმარიტი	ჭეშმარიტი	მცდარი	ჭეშმარიტი	ჭეშმარიტი

ლოგიკური ოპერატორი აერთიანებს დასრულებულ ლოგიკურ გამოსახულებებს. მაგალითად a > b & b > c ვარგისი გამოსახულებაა მაგრამ a > b & c არ არის მისი ექვივალენტური.

ლოგოკური გამოსახულება შეიძლება იწყებოდეს ~ ნიშნით. ეს ოპერატორი გამოსახულების მნიშვნელობას მისი საპირისპიროთი ცვლის. თუ a > b ჭეშმარიტია, მაშინ ~ (a > b) – მცდარია.

ლოგიკური გამოსახულება შესაძლოა შეიცავდეს რამდენიმე ლოგიკურ ოპერატორს:

```
~ (b == c | b == 5.5)
```

ლოგიკური ოპერატორების იერარქია (ზემოდან ქვემოთ) ასეთია: არ, და, ან. იერარქიის შესაცვლელად ფრჩხილები უნდა გამოვიყენოთ. ზემოთ მოცემულ გამოსახულებაში პირველ რიგში შესრულდება b == c და b == 5.5. დავუშვათ b = 3 c = 5. რადგან არცერთი გამოსახულება არ არის ჭეშმარიტი, გამოსახულება - b == c | b == 5.5 მცდარია, ფრჩხილების წინ მდგარი ~ ოპერატორი კი მას საპირისპირო მნიშვნელობას – ჭეშმარიტს მიანიჭებს. დავუშვათ ფრჩხილები არ გვაქვს ამ გამოსახულებაში:

~ b == c | b == 5.5

ამ შემთხვევაში გამოსახულება ~ b == c შეფასდება b == 5.5 გამოსახულებასთან ერთად . b და c მოცემულ მნიშვნელობათათვის ორივე გამოსახულება მცდარია, მთლიანად გამოსახულების მნიშვნელობაც მცდარია. შეიძლება გაგიჩნდათ კითხვა, როგორ შევაფასეთ ~b, როცა b თვითონ რიცხვს წარმოადგენს. MATLAB –ში ყველა არანულოვანი მნიშვნელობა შეფასდება როგორც ჭეშმარიტი, ხოლო ნოლოვანი მნიშვნელობა შეფასდება როგორც მცდარი. ამიტომ დიდი სიფრთხილეა საჭირო შედარების და ლოგიკური ოპერატორებით სარგებლობისას, რათა პროგრამის ყველა საფეხური ისე შესრულდეს, როგორც ამას ჩვენი ამოცანა მოითხოვს.

სავარჯიშო

განსაზღვრეთ ჭეშმარიტია თუ მცდარი შემდეგი გამოსახულებები. შეამოწმეთ პასუხები MATLAB საშუალებით. გაიხსენეთ, რომ პასუხის შესამოწმებლად საჭიროა ცვლადებისათვის მნიშვნელობათა მინიჭება და შემდეგ გამოსახულების შეყვანა კლავიატურიდან ბრძანებათა ფანჯარაში:

a = 5.5; b = 1.5; k = -3;

1. a < 10.0
2. a + b >= 6.5
3. k ~= 0
4. b – k > a
5. ~ (a == 3* b)
6. – k <= k + 6
7. a < 10 & a > 5
8. abs (k) > 3 | k < b – a

მარტივი if ბრძანება

პროგრამაში **if** ბრძანება შეიძლება გამოვიყენოთ მარტივი ფორმით, მხოლოდ if ბრძანება, ანდა მისი რთული ფორმა, რომელიც გარდა **if** ბრძანებისა, შეიცავს **else** და **elseif** ოპერატორებს. განვიხილოთ მარტივი ფორმა ამ ბრძანებისა, რომელსაც აქვს ასეთი სახე:

```
if ლოგიკური გამოსახულება
    ბრძანებათა ჯგუფი A
end
```

თუ ლოგიკური გამოსახულება ჭეშმარიტია, სრულდება ბრძანებები **if** და **end** შორის, ხოლო თუ ლოგიკური გამოსახულება მცდარია, ბრძანებათა ჯგუფი A გამოტოვებული იქნება და პროგრამა გადავა იმ ბრძანებათა შესრულებაზე, რომელიც **end** –ს მოჰყვება. პროგრამის დაწერისას იმისათვის, რომ გამოიკვეთოს **if** ბრძანებით გათვალისწინებული საფეხური, უკეთესია ბრძანებათა ეს ჯგუფი აბზაცით დაიწეროს. მაგალითად:

```
if a < 50
    count = count + 1;
    sum = sum +a;
end
```

დავუშვათ a სკალარია. თუ a < 50, ცვლადი count გაიზრდება ერთით და sum ცვლადის მნიშვნელობას დაემატება a მნიშვნელობა, თუ არა, პროგრამა გამოტოვებს ამ ორ ბრძანებას. თუ a სკალარი არ არის, ორივე ბრძანება შესრულდება მხოლოდ მაშინ, თუ a-ს ყველა ელემენტი < 50.

if ბრძანება შეიძლება იყოს ერთმანეთში ჩასმული:

```
if ლოგიკური გამოსახულება 1
    ბრძანებათა ჯგუფი A
    if ლოგიკური გამოსახულება 2
       ბრძანებათა ჯგუფი B
    end
    ბრძანებათა ჯგუფი C
end
ბრძანებათა ჯგუფი D
```

თუ 'ლოგიკური გამოსახულება 1' ჭეშმარიტია, შესრულდება ბრძანებათა ჯგუფი A და C, თუ 'ლოგიკური გამოსახულება 2' ასევე ჭეშმარიტია, შესრულდება ბრძანებათა ჯგუფი B-ც, ვიდრე შესრულდება ბრძანებათა ჯგუფი C. თუ 'ლოგიკური გამოსახულება 1' მცდარია, პროგრამა პირდაპირ გადადის ბრძანებათა D ჯგუფზე. აბზაცების გამოყენება **if** ბრძანების ასეთ სტრუქტურაში ძალზე მნიშვნელოვანია.
განვიხილოთ მაგალითი, რომელიც შეიცავს ერთმანეთში ჩასმულ **if** ბრძანებებს.

```
if a < 50
    count = count + 1;
    sum = sum +a;
    if b > a
       b = 0;
    end
end
```

დავუშვათ a და b სკალარებია, მაშინ როცა a < 50, ცვლადი count გაიზრდება 1 –ით და ცვლადს sum დაემატება a. გარდა ამისა, თუ b > a მაშინ b მიენიჭება მნიშვნელობა 0. თუ a არ არის 50-ზე ნაკლები, მაშინ პროგრამა პირდაპირ გადადის ბრძანებაზე, რომელიც მოჰყვება მეორე end –ს. თუ a სკალარი არ არის, მაშინ პირობა a < 50 მხოლოდ მაშინაა ჭეშმარიტი, როცა იგი სრულდება a-ს ყველა მნიშვნელობისათვის. თუ არც a და არც b არ არის სკალარი, მაშინ b მეტია a-ზე მხოლოდ მაშინ, როცა b ყველა ელემენტი მეტია a შესაბამის ელემენტზე. თუ a ან b სკალარია, მაშინ მატრიცის ელემენტები შედარდება სათითაოდ სკალარს და ისე შემოწმდება პირობა.

შედარების და ლოგიკური ფუნქციები

MATLAB აქვს შედარების და ლოგიკური ფუნქციები, რომლებიც გამოიყენება **if** ბრძანების სტრუქტურაში.

any (x)	x მატრიცის ყოველი სვეტისათვის გვაძლევს 1, თუ ამ სვეტის რომელიმე ელემენტი მაინც არის არანულოვანი, სხვა შემთხვევაში გვაძლევს 0
all (x)	x მატრიცის-ის ყოველი სვეტისათვის გვაძლევს 1, თუ ამ სვეტის ყველა ელემენტი არანულოვანია, სხვა შემთხვევაში გვაძლევს 0.
find (x)	გვაძლევს ვექტორს, რომელიც შეიცავს x ვექტორის არანულოვანი ელემენტების ინდექსებს. თუ x მატრიცაა, მაშინ იდექსები შეირჩევა x(:) ვექტორიდან. ეს არის ერთი გრძელი ვექტორი, რომელიც შედგება x მატრიცის ერთმანეთს მიყოლებული სვეტების ელემენტებისაგან.
exist ('A')	გვაძლევს 1, თუ მოცემულ სამუშაო სივრცეში არსებობს A ცვლადი , 2 თუ მოცემულ სამუშაო სივცეში არსებობს A ცვლადი ან A.m ფაილი, გვაძლევს 0 თუ ასეთი არც ცვლადი და არც .m ფაილი არ არსებობს. (ყურადღება მიაქციეთ: სახელი ბრჭყალებში უნდა იყოს ჩასმული)
isnan(x)	გვაძლევს მატრიცას, რომლის ელემენტები უდრის 1, როცა x მატრიცის შესაბამისი ელემენტები წარმოადგენს განუზღვრელობას, სხვა შემთხვევაში - 0.
finite (x)	გვაძლევს მატრიცას, რომლის ელემენტები ტოლია 1, როცა

	x მატრიცის შესაბამისი ელემენტები სასრული რიცხვებია, სხვა შემთხვევაში – 0.
isepty (x)	გვაძლევს 1, თუ x არის ცარიელი მატრიცა, თუ არა - 0.
isstr (x)	გვაძლევს 1, თუ x სიმბოლოებისგან შემდგარი სტრიქონია, 0 - თუ ეს ასე არაა
strcmp(y1,y2)	ერთმანეთს შეადარებს ორ სტრიქონს y1 და y2 და გვაძლევს 1, თუ ისინი იდენტურია, 0 – თუ ისინი განსხვავებულია.

დავუშვათ A სამსტრიქონიანი და სამსვეტიანი მატრიცაა. განვიხილოთ შემდეგი ბრძანება:

```
if all (A) == 0
     disp ('A contains all zeros')
end
```

სტრიქონი A contains all zeros დაიბეჭდება მხოლოდ იმ შემთხვევაში, როცა A ყველა ელემენტის მნიშვნელობა არის 0 –ის ტოლია.

სავარჯიშო

განსაზღვრეთ ყოველი გამოსახულების მნიშვნელობა. შეამოწმეთ პასუხები MATLAB საშუალებით. დავუშვათ მატრიცა B –ს აქვს მნიშვნელობა:

$$B = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 0 & 3 \\ 8 & 7 & 0 \end{bmatrix}$$

1. any (B)
2. find (B)
3. all (any (B))
4. any (all (B))
5. finite (B(:,3))
6. any (B(1:2,1:3))

else ოპერატორი

ეს ოპერატორი საშუალებას გვაძლევს გავუშვათ ბრძანებათა ერთი ჯგუფი, როცა ლოგიკური გამოსახულება ჭეშმარიტია, და ბრძანებათა სხვა ჯგუფი, როცა იგივე გამოსახულება მცდარია. **else** ოპერაორის შემცველი if ბრძანების ზოგადი ფორმაა:

```
if  ლოგიკური გამოსახულება
    ბრძანებათა ჯგუფი A
else
    ბრძანებათა ჯგუფი B
end
```

თუ 'ლოგიკური გამოსახულება' ჭეშმარიტია, სრულდება ბრძანებათა ჯგუფი A, თუ მცდარია, სრულდება ბრძანებათა ჯგუფი B. **else** შეიძლება გამოვიყენოთ ერთმანეთში ჩასმულ **if** ბრძანებებშიც.

საილუსტრაციოდ განვიხილოთ მაგალითი:

ბაგირგზა აკავშირებს ორ კოშკს. დავუშვათ სკალარი d წარმოადგენს მანძილის მნიშვნელობას ვაგონიდან უახლოეს კოშკამდე. თუ ვაგონი კოშკიდან 30-მდე ფუტითაა დაშორებული, სიჩქარე გამოითვლება ფორმულით:

$$velocity = 0.425 + 0.00175d^2$$

თუ ვაგონი კოშკიდან უფრო შორსაა, ვიყენებთ ფორმულას:

$$velocity = 0.625 + 0.12d + 0.00025d^2$$

სიჩქარის მნიშვნელობები შეგვიძლია გამოვითვალოთ შემდეგი ბრძანების საშუალებით:

```
if d < 30
   velocity = 0.425 + 0.00175*d^2;
   else
   velocity = 0.625 + 0.12*d + 0.00025*d^2;
end
```

elseif ოპერატორი

elseif ოპერატორს მივმართავთ, იმისათვის რომ შევამოწმოთ რამდენიმე ლოგიკური გამოსახულება და გამოთვლები საჭირო მიმართულებით წარვმართოთ:

```
if   ლოგიკური გამოსახულება 1
       ბრძანებათა ჯგუფი A
elseif ლოგიკური გამოსახულება 2
       ბრძანებათა ჯგუფი B
elseif ლოგიკური გამოსახულება 3
       ბრძანებათა ჯგუფი C
end
```

გვაქვს ორი **elseif** ოპერატორი, შეგვიძლია უფრო მეტიც გამოვიყენოთ. თუ 'ლოგიკური გამოსახულება 1' ჭეშმარიტია, მხოლოდ A ჯგუფის ბრძანებები სრულდება. თუ 'ლოგიკური გამოსახულება 1' მცდარია და 'ლოგიკური გამოსახულება 2' ჭეშმარიტია, მხოლოდ ბრძანებათა ჯგუფი B სრულდება, თუ 'ლოგიკური გამოსახულება 1' და 'ლოგიკური გამოსახულება 2' მცდარია და 'ლოგიკური გამოსახულება 3' ჭეშმარიტია, მხოლოდ ბრძანებათა ჯგუფი C სრულდება. თუ ერთზე მეტი ლოგიკური გამოსახულებაა ჭეშმარიტი, პირველი ჭეშმარიტი გამოსახულება განსაზღვრავს, თუ ბრძანებათა რომელი ჯგუფი უნდა შესრულდეს. თუ არც ერთი ლოგიკური გამოსახულება არ არის ჭეშმარიტი, **if** სტრუქტურის არცერთი ბრძანება არ შესრულდება.

შეიძლება **if** სტრუქტურაში **else** და **elseif** ოპერატორები ერთდროულად გამოვიყენოთ:

```
if   ლოგიკური გამოსახულება 1
       ბრძანებათა ჯგუფი A
elseif ლოგიკური გამოსახულება 2
       ბრძანებათა ჯგუფი B
elseif ლოგიკური გამოსახულება 3
```

```
    ბრძანებათა ჯგუფი C
else
    ბრძანებათა ჯგუფი D
end
```

თუ არცერთი ლოგიკური გამისახულება არ არის ჭეშმარიტი, მაშინ შესრულდება ბრძანებათა ჯგუფი D. if-elseif სტრუქტურას ხშირად ამომრჩევ სტრუქტურასაც უწოდებენ, რადგანაც მოწმდება რამდენიმე პირობა.

სავარჯიშო

დაწერეთ MATLAB ბრძანებების მწკრივი, რომ შესრულდეს განსაზღვრული საფეხურები, დავუშვათ ცვლადები საკალარული სიდიდეებია.

1. თუ დრო მეტია, ვიდრე 50, დაუმატე მას 1.
2. როცა კვადრატული ფესვი ცვლადიდან poly ნაკლებია ვიდრე 0.001, დაბეჭდე მისი მნიშვნელობა
3. თუ სხვაობა volt_1 და volt_2 შორის მეტია, ვიდრე 2.0, დაბეჭდე მათი მნიშვნელობები
4. თუ den მნიშვნელობა ნაკლებია ვიდრე 0.003, მიანიჭეთ result –ს 0-ის ტოილ მნიშვნელობა, თუ არა, მიანიჭეთ result-ს მნიშვნელობა num/den
5. თუ x-ის ლოგარითმი ნატურალური ფუძით მეტია, ან ტოლია 10-ის, მიანიჭეთ time –ს 0-ის ტოილ მნიშვნელობა და გაზარდეთ count მნიშვნელობა
6. თუ dist ნაკლებია 50, და time მეტია 10, გაზარდეთ time მნიშვნელობა 2-ით, თუ არა, გაზარდეთ time 5-ით
7. თუ dist მეტია ან ტოლია 100, გაზარდეთ time 10-ით, თუ dist მოთავსებულია 50 და 100 შორის, გაზარდეთ time 1-ით. სხვა სემთხვევაში გაზარდეთ time 0.5-ით

for cikli

ციკლისათვის MATLAB აქვს ორი ოპერატორი **for** და **while**. ჯერ განვიხილავთ ციკლის ოპერატორს **- for**.

მას აქვს ზოგადი სახე:

```
for index = გამოსახულება
    ბრძანებათა ჯგუფი A
end
```

გამოსახულება შეიძლება იყოს მატრიცა, ვექტორი ან სკალარი. ბრძანებათა ჯგუფი სრულდება იმდენჯერ, რამდენი სვეტიცაა მატრიცაში. ინდექსი მიმდევრობით ღებულობს მატრიცის ელემენტების მნიშვნელობებს. **for** ოპერატორის გაცნობამდე განვიხილოთ მაგალითი:

დავუშვათ გვაქვს ვექტორი, რომელიც შეიცავს მანძილების მნიშვნელობებს, რომლებიც შეესაბამება საბაგირო გზის ვაგონის დაშორებას უახლოესი კოშკიდან. გვინდა მივიღოთ ვექტორი, რომელიც შეიცავს აჩქარების შესაბამის მნიშვნელობებს. თუ მანძილი ნაკლებია 30 ფუტზე, უნდა ვისრგებლოთ ფორმულით:

$$velocity = 0.425 + 0.00175d^2$$

თუ ვაგონი კოშკიდან უფრო შორსაა, ვიყენებთ ფორმულას:

$$velocity = 0.625 + 0.12d + 0.00025d^2$$

სიჩქარის ფორმულის არჩევა დამოკიდებულია d მაძილის მნიშვნელობაზე. იმისათვის, რომ d თითოეული მნიშვნელობისათვის ცალკ-ცალკე არ გამოვითვალოთ სიჩქარეები, ვიყენებთ **for** ოპერატორს:

დავუშვათ d ვექტორი შეიცავს 25 ელემენტს.

```
for k = 1:25
    if d(k) <= 30
        velocity = 0.425 + 0.00175*d(k)^2;
    else
        velocity = 0.625 + 0.12*d(k) + 0.00025*d(k)^2;
    end
end
```

შემდეგ ამოხსნაში დავუშვებთ, რომ d ვექტორის ზომა უცნობია. ამიტომ ვისრგებლებთ **length** ბრძანებით.

```
for k = 1:length(d)
    if d(k) <= 30
        velocity = 0.425 + 0.00175*d(k)^2;
    else
        velocity = 0.625 + 0.12*d(k) + 0.00025*d(k)^2;
    end
end
```

for ოპერატორით სარგებლობისას უნდა დავიცვათ შემდეგი წესები:

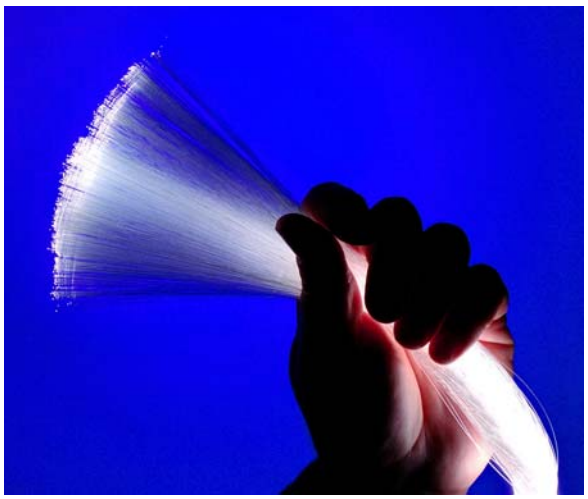
1. **for** ოპერატორის ინდექსი უნდა იყოს ცვლადი
2. თუ გამოსახულება ცარიელი მატრიცაა, ციკლის გაშვება არ მოხდება, პროგრამის კონტროლი გამოტოვებს ყველა ბრძანებას, რომელიც მოთავსებულია **for** და **end** შორის.
3. თუ გამოსახულება სკალარია, ციკლი მხოლოდ ერთხელ შესრულდება
4. თუ გამოსახულება სტრიქონი ვექტორია, ინდექსი მორიგეობით ღებულობს ვექტორის ელემენტების მნიშვნელობებს.
5. თუ გამოსახულება მატრიცაა, ინდექსი მიიღებს მნიშვნელობებს ჯერ პირველი სვეტის ელემენტებისას, შემდეგ მეორე სვეტის და ა. შ.
6. ციკლის ოპერატორის დასრულებისას ინდექსი ინარჩუნებს საბოლოოდ მიღებულ მნიშვნელობას
7. თუ გამოსახულების მატრიცის მისაღებად გამოყენებულია ორწერტილიანი ოპერატორი
   for k = საწყისი მნიშვნელობა : ნაზრდი : საბოლოო მნიშვნელობა
   მაშინ იმის დასადგენად, თუ რამდენჯერ შესრულდება ციკის ოპერატორით განსაზღვრული მოქმედებები, ვისარგებლებთ ფორმულით:
   floor((საბოლოო მნიშვნელობა – საწყისი მნიშვნელობა)/ნაზრდი)+1
   თუ ეს სიდიდე უარყოფითია, ციკლის გაშვება არ მოხდება.

განსაზღვრეთ რამდენჯერ შესრულდება ციკლის ოპერატორით განსაზღვრული მოქმედება თუ ინდექსის გამოსახულება შემდეგი სახისაა. შეამოწმეთ პასუხი MATLAB საშუალებით.

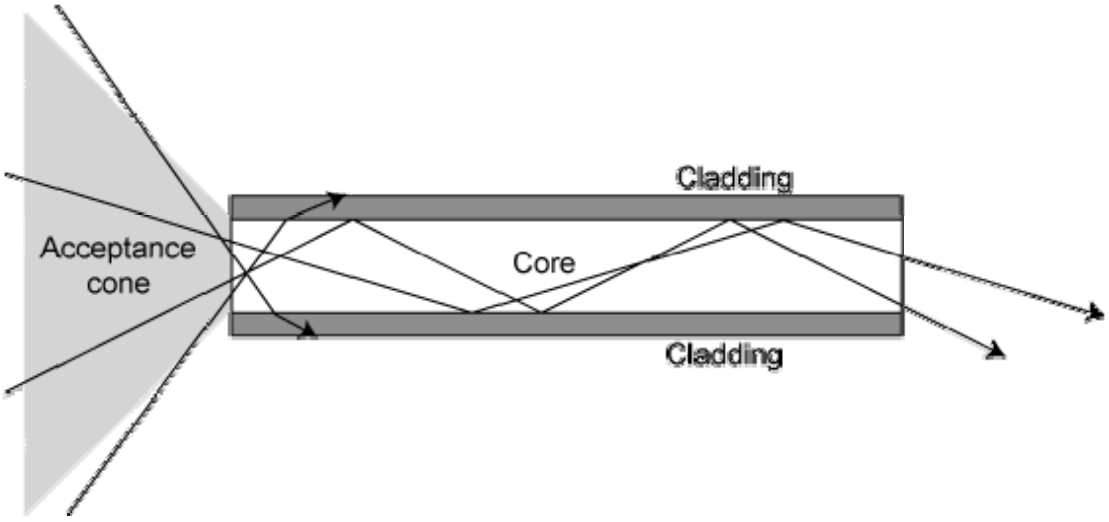
9. for k = 3:20
10. for count = -2:14
11. for k = -2:-1:-10
12. for time = 10:5
13. for index = 52:-12

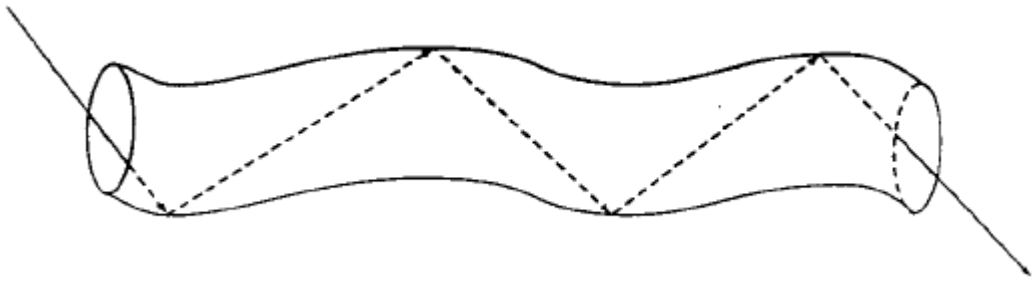
ციკლი შეგვიძლია შევწყვიტოთ ბრძანებით: **break.** ეს ბრძანება აუცილებლად დაგვჭირდება, თუ ციკლის ოპერატორში პირობა შეცდომითაა განსაზღვრული.

პრობლემა: ოპტიკური ბოჭკო



ნახ. 4.1 ოპტიკურ ბოჭკოთა კონა





ნახ. 4.2 ოპტიკური ბოჭკო

თუ სინათლე ეცემა მინის ან პლასტიკის გრძელ ღეროს, იგი მრავალგზის აირეკლება ღეროს კედლებიდან ვიდრე მეორე ბოლოს მოაღწევს. ეს საინტერესო ფენომენი შეიძლება გამოყენებული იქნას სინათლის ან სულაც გამონასახის გადასაცემად. თუ ავიღებთ ასეთი ღეროების კონას, სინათლე გავრცელდება მასში და მიაღწევს მეორე ბოლოს, როგორც ეს ნახ. 4.2 მოცემული.

ოპტიკური ბოჭკო არის ძალიან წვრილი მინის ან პლასტიკის ბოჭკო. ასეთი ბოჭკოების კონა შეიძლება გამოვიყენოთ როგორც სინათლის გადაცემის საშუალება. ოპტიკური ბოჭკო გამონასახის გადაცემის საშუალებასაც იძლევა, თუ ბოლოებზე სპეციალურ მოწყობილობას მივუერთებთ. რადგან ოპტიკური ბოჭკო ძალზე მოქნილია, ის შეგვიძლია გამოვიყენოთ როგორც გამონასახის გადაცემის საშუალება ისეთ ადგილებში, რომელიც სხვა ხელსაწყოებისათვის მიუწვდომელია, მაგალითად ენდოსკოპში. ენდოსკოპი ხელსაწყოა, რომელიც გამოიყენება პაციენტის ორგანიზმის გამოსაკვლევად. ორგანიზმში შეღწევა ხდება უმცირესი ჭრილობის საშუალებით. ოპტიკური ბოჭკო ლაზერული სხივის გადასაცემადაც გამოიყენება, რომელიც გამოიყენება არტერიების გასათავისუფლებად, თირკმელში ქვების დასამსხვრევად, კატარაქტის მოსაცილებლად და სხვა.

ბოჭკოებში შიდა არეკვლის ფენომენი აიხსნება შნელის კანონით. ოპტიკური ბოჭკო შედგება ორგვარი მასალისაგან – მასალა, რომლისგანაც შედგება ბოჭკო და მასალა, რომელიც გარს აკრავს მას. ჩვეულებრივ ბოჭკოს შემადგენელი მასალა უფრო მკვრივია, ვიდრე გარსი. როცა სინათლე ერთი გარემოდან განსხვავებული სიმკვრივის მქონე მეორე გარემოში გადადის, იგი გარდატყდება გამყოფ ზედაპირზე. გარდატეხის კუთხე დამოკიდებულია გარემოს გარდატეხის კოეფიციენტზე და დაცემის კუთხეზე. როცა სინათლე უფრო მკვრივიდან ნაკლებ მკვრივ გარემოში გადადის, დაეცემა რა ზედაპის მისი ნაწილი აირეკლება, ნაწილი კი გააღწევს მეორე გარემოში. სინათლის დაცემის კუთხეს, როცა იგი მთლიანად აირეკლება ზედაპირიდან კრიტიკული კუთხე ეწოდება. რადგანაც კრიტიკული კუთხე დამოკიდებულია ერთი გარემოს მეორის მიმართ გარდატეხის კოეფიციენტზე შეგვიძლია გამოვითვალოთ ეს კუთხე და განვსაზღვროთ მოცემული კუთხით დაცემული სინათლე გაივლის თუ არა ბოჭკოში. დავუშვათ n_2 გარემომცველი გარემოს გარდატეხის კოეფიციენტია, ხოლო n_1 თავად ბოჭკოსი, თუ $n_2 > n_1$ ბოჭკო სინათლეს არ გაატარებს. კრიტიკული კუთხე გამოითვლება ფორმულით:

$$\sin\theta_c = \frac{n_2}{n_1}$$

დაწერეთ MATLAB პროგრამა, რომელიც განსაზღვრავს გაივლის თუ არა სინათლე ოპტიკურ ბოჭკოში, რომელიც გარემოცულია განსხვავებული სიმკვრივის მქონე მასალით. დავუშვათ, გვაქვს ASCII მონაცემთა ფაილი სახელით indicies.dat, რომელიც შეიცავს

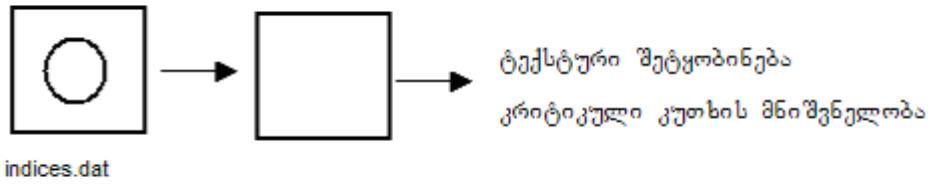
მონაცემებს ოპტიკური ბოჭკოსა და გარემომცველი გარემოს შესახებ. ფაილის ყოველი სტრიქონი შეიცავს ბოჭკოსა და გარსის მასალის გარდატეხის კოეფიციენტს. პროგრამამ უნდა განსაზღვროს გაატარებს თუ არა სინათლეს ასეთი მონაცემების მქონე გამტარი და თუ გაატარებს, როგორია კრიტიკული კუთხის მნიშვნელობა.

1. ამოცანის დასმა

განვსაზღვროთ, გაივლის თუ არა სინათლე მოცემულ გარემოში. განვსაზღვროთ დაცემის კუთხე, როცა სინათლე გაივლის ბოჭკოში.

2. INPUT/OUTPUT აღწერა

როგორც ნაჩვენებია 4.2 სურათზე, პროგრამის შესავალი მნიშვნელობაა მონაცემთა ფაილი, რომელიც შეიცავს პოტენციური სინათლის გამტარის გარდატეხის კოეფიციენტებს. გამოსავალი მნიშვნელობა კი ცნობა იმის შესახებ გაივლის თუ არა სინათლე მოცემული პარამეტრების მქონე გამტარში და კრიტიკული კუთხე.



ნახ. 4.3 I UT/OUTPUT დიაგრამა

3. სახელდახელო ამოხსნა

ჰაერის გარდატეხის კოეფიციენტია 1.0003, მინის – 1.5. თუ შევქმნით სინათლის გამტარს ჰაერით გარემოცული მინის ბოჭკოთი, კრიტიკული კუთხე შეიძლება გამოვითვალოთ ფორმულით:

$$\theta_c = \sin^{-1}\frac{n_2}{n_1} = \sin^{-1}\frac{1.0003}{1.5} = \sin^{-1}(0.66687) = 41.82^o$$

ასეთი გამტრი გაატარებს სინათლეს, რომელიც დაეცემა 41.82^o-ზე მეტი კუთხით.

4. MATLAB ამოხსნა

```
%
%     This program reads the indices of  refraction
%     for materials forming a light pipe. For each
%     pair of materials, we determine if the pipe
%     will transmit light, and at what angles of
%     incidence in degrees.
%
factor = 180/pi;
```

```
load indices.dat
n1 = indices(:,1);
n2 = indices(:,2);
for k = 1: length(n1)
    if  n2(k) > n1(k)
        fprintf('light is not transmitted for \n')
        fprintf('rod  index  %g  and  medium  index  %g  \n\n',
            n1(k), n2(k))
    else
        critical_angle = asin(n2(k)/n1(k))*factor;
        fprintf('light is not transmitted for \n')
        fprintf('rod  index  %g  and  medium  index  %g  \n\n',
            n1(k), n2(k))
        fprintf('for angles greater then %g  degrees \n\n',…
            critical_angle)
    end
end
```

5. შემოწმება

შევამოწმოთ ეს ამოცანა ფაილით რომლის შემცველობაა:

1.31	1.473
1.5	1.0003
1.49	1.33

მივიღებთ:

```
light is not transmitted for
rod index 1.31 and medium index 1.473
light is transmitted for
rod index 1.5 and medium index 1.0003
for angles greater then 41.8257  degrees
light is transmitted for
rod index 1.49 and medium index 1.33
for angles greater then 63.204  degrees
```

სცადეთ იპოვოთ მასალები, რომელთათვისაც დაცემის კრიტიკული კუთხე 45 მეტია.

while cikli

ეს ბრძანება მნიშვნელოვანია, როცა გვსურს პროგრამაში მოქმედებები განმეორდეს ვიდრე გარკვეული პირობა სრულდება. მისი ზოგადი ფორმაა:

while გამოსახულება
    ბრძანებათა ჯგუფი A
end

თუ გამოსახულება ჭეშმარიტია, A ჯგუფის ბრძანებები სრულდება, ამის შემდეგ მოწმდება პირობა ხელახლა. თუ პირობა კვლავ ჭეშმარიტია, ისევ სრულდება ბრძანებათა A ჯგუფი და ასე გრძელდება, ვიდრე პირობა არ გახდება მცდარი. ამის შემდეგ პროგრამა გადადის **end** –ის მომდევნო ბრძანებაზე. ცვლადები, რომლებიც მონაწილეობენ ბრძანებათა ჯგუფში, უნდა შეიცავდნენ გამოსახულებაში მონაწილე ცვლადებს. თუ ეს ასე არ არის,

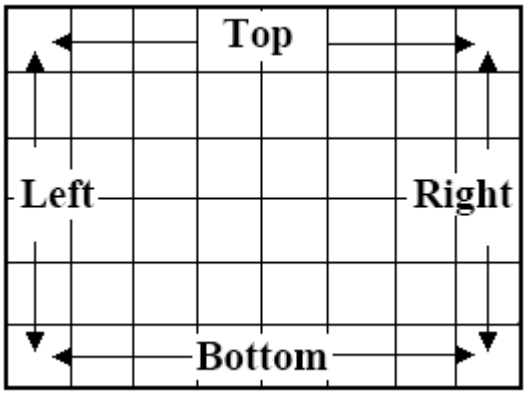
გამოსახულების მნიშვნელობა არ შეიცვლება. თუ გამოსახულება ყოველთვის ჭეშმარიტი რჩება(ან არის არანულოვანი რიცხვი), ციკლი უსასრულოდ გრძელდება. (გაიხსენეთ, რომ ციკლის შეწყვეტა შეგიძლიათ ბრძანებით **^c (ctrl+c).**

while ციკლის საილუსტრაციოდ განვიხილოთ ბრძანება, რომელიც ვექტორის ელემენტებს უმატებს ცვლადს sum, ვიდრე უარყოფითი რიცხვი არ შეხვდება.

```
sum = 0;
k = 1;
while x(k) >= 0 & k <= length(x)
    sum = sum + x(k);
    k = k + 1;
end
```

პრობლემა – ტემპერატურული წონასწორობა

ახალი მასალების გამოგონება და გაუმჯობესება საჭიროებს მათ გამოცდას მრავალი პარამეტრის მიხედვით. ერთ-ერთი მათგანია ტემპერატურის განაწილება. ამ ამოცანაში განვიხილავთ ტემპერატურის განაწილებას მეტალის თხელ ფირფიტაზე. ფირფიტის მასალა შექმნილია ისეთი მოწყობილობისათვის, რომელშიც მისმა ოთხივე კიდემ უნდა შეინარჩუნოს მუდმივი, ან იზოთერმული ტემპერატურა. ფირფიტის სხვა წერტილებში ტემპერატურა გამოითვლება მეზობელი წერტილების ტემპერატურის მიხედვით. წარმოვიდგინოთ, რომ ფირფიტა დაფარულია ბადით, მაშინ ფირფიტა შეგვიძლია განვიხილოთ, როგორც მატრიცა, რომლის ელემენტები წარმოადგენს ტემპერატურას შესაბამის უჯრედში. ნახ. 4.4 წარმოადგენს ამგვარი ბადით დაფარულ ფირფიტას. ჩავთვალოთ, რომ ბადის უჯრედის შიგნით ტემპერატურა მუდმივია. ფირფიტა დაყოფილია 6 × 9 უჯრედად. სულ ტემპერატურის 48 მნიშვნელობა გვაქვს მოცემული მატრიცის სახით.



ნახ. 4.4 მეტალის ფირფიტის დაყოფა

მოცემულია იზოთერმული ტემპერატურა ფირფიტის ოთხივე გვერდისთვის; დავუშვათ ფირფიტის გვერდითი და ზედა კიდეები ერთნაირი ტემპერატურისაა, ხოლო ქვედა კიდე განსხვავებულია. იგულისხმება, რომ დასაწყისისათვის ფირფიტის სხვა უჯრედების

ტემპერატურა 0-ის ტოლია. ყოველი შიდა წერტილისათვის ახალი ტემპერატურა გამოითვლება როგორც მისი მეზობელი წერტილების საშუალო შემდეგი განტოლებით:

	T_1	
T_4	T_0	T_2
	T_3	

$$T_0 = \frac{T_1 + T_2 + T_3 + T_4}{4}$$

შიდა წერტილებისათვის ახალი ტემპერატურების გამოთვლის შემდეგ შეფასდება სხვაობა ტემპერატურის ძველსა და ახალ მნიშვნელობას შორის. თუ ტემპერატურის ცვლილება მეტია, ვიდრე წინასწარ განსაზღვრულ სიდიდეზე – ფირფიტა ჯერ კიდევ არ არის თერმულ წონასწორობაში და აღწერილი პროცესი ისევ განმეორდება.

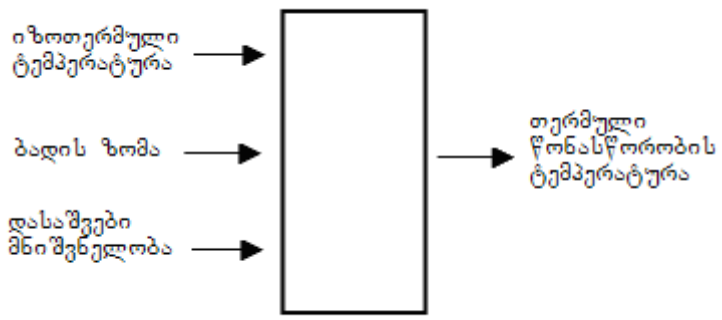
ვსარგებლობთ ორი მატრიცით ტემპერატურებისათვის; ერთი ძველი მნიშვნელობებისათვის, მეორე ახლად გამოთვლილი ტემპერატურისათვის. ორი მატრიცა იმისათვის გვჭირდება, რომ ვუშვებთ: ტემპერატურის ცვლილება ყველა ელემენტისათვის ერთდროულად ხდება. მაგალითად ვითვლით ტემპერატურას წერტილში (3,3). ახალი მნიშვნელობა გამოითვლება როგორც საშუალო მისი მეზობელი მნიშვნელობებისა (2,3), (3,2), (3,4) და (4,3). შემდეგ ვითვლით ტემპერატურას მდებარეობისათვის (3,4). ისევ ვითვლით საშუალოს, მაგრამ ჩვენ გვჭირდება (3,3) მდებარეობის ძველი და არა ახლად გამოთვლილი მნიშვნელობა, ამიტომ საჭიროა ტემპერატურის ახლადგამოთვლილი მნიშვნელობები ჩავწეროთ ახალი მატრიცის სახით.

ამრიგად, ვიყენებთ მატრიცას ძველი ტემპერატურებით იმისათვის, რომ გამოვითვალოთ ახალი ტემპერატურები და ვამოწმებთ თუნდაც ერთი სხვაობა ხომ არ აჭარბებს დასაშვებ მნიშვნელობას. როცა ტემპერატურის ცვლილება ფირფიტის ყოველი უჯრედისათვის, ე. ი ნებისმიერი სხვაობა ახალი და ძველი მატრიცის შესაბამის ელემენტებს შორის გახდება დასაშვებ მნიშვნელობაზე ნაკლები, ჩავთვლით, რომ თერმული წონასწორობა დამყარდა.

1. ამოცანის დასმა

განვსაზღვროთ თერმული წონასწორობის შესაბამისი სიდიდეები მეტალის ფირფიტისათვის იზოთერმული კიდეებით.

2. INPUT/OUTPUT აღწერა



ნახ. 4.5 I/O დიაგრამა

როგორც ნახაზზეა ნაჩვენები, პროგრამის შესავალი მნიშვნელობაა ფირფიტის ბადის ზომა, იზოთერმული ტემპერატურა და დასაშვები მნიშვნელობა.

3. სახელდახელო ამოხსნა

რომ დავრწმუნდეთ რამდენად კარგად გავიგეთ აღწერილი პროცესი, ამოვხსნათ პრობლება მარტივი შემთხვევისათვის. დავუშვათ მატრიცა შეიცავს 4 სტრიქონსა და 4 სვეტს. გვერდითი და ზედა კიდეების იზოთერმული ტემპერატურა 100 გრადუსია, ხოლო ქვედა კიდისათვის – 50 გრადუსი. დასაშვები სიდიდე ავიღოთ 40 გრადუსის ტოლი. ფირფიტის ელემენტების ტემპერატურის საწყისი მნიშვნელობები იქნება:

100 100 100 100
100 0 0 100
100 0 0 100
50 50 50 50

პირველი იტერაციის შემდეგ მივიღებთ;

100 100 100 100
100 50 50 100
100 37.5 037.5 100
50 50 50 50

მეორე იტერაციის შემდეგ მივიღებთ:

100 100 100 100
100 71.875 071.875 100
100 059.375 059.375 100
50 50 50 50

ამის შემდეგ ტემპერატურათა სხვაობა არ აღემატება 40 გრადუსს ფირფიტის არცერთი წერტილისათვის. ესე იგი თერმული წონაწორობა დამყარდა და მეორე იტერაციის შედეგად მიღებული ტემპერატურები შეესაბამება ტემპერატურულ წონასწორობას.

4. MATLAB ამოხსნა

```
%     This program initializes the temperatures in
%     metal plate and determines the equilibruim
%     temperatures based on atolerance value
%
nrows = input(' Enter number of rows');
ncols = input('Enter number of columns');
iso1 = input('Enter temperature for top and sides');
iso2 = input('Enter temperature for bottom');
tolerance = input('Enter equilibrium tolerance');
%     %
%     %     Initialize and print temperature matrix
%     %
old = zeros(nrows, ncols);
old(1,:) = iso1 + zeros(1, ncols);
```

```
old(:,1) = iso1 + zeros(nrows,1);
old(:,ncols) = iso1 + zeros(nrows,1);
old(nrows,:) = iso2 + zeros(1, ncols);
disp('Initial Temperatures');
disp(old)
new = old;
equilibrium = 0;
%    %
%    %     Update temperatures and test for equilibrium
%    %
while ~equilibrium
    for m = 2:nrows - 1
        for n = 2: ncols - 1
            new(m,n) = (old(m-1,n)+old(m,n-1)+...
            old(m,n+1)+old(m+1,n))/4;
        end
    end
    if all(new-old<=tolerance)
        equilibrium = 1;
        disp('Equilibrium Temperatures');
        disp(new)
    end
    old = new;
end
```

5. შემოწმება

თუ გავუშვებთ პროგრამას MATLAB –ში და მივაწოდებთ სათანადო მნიშვნელობებს, შედეგად მივიღებთ:

```
Enter number of rows  4
Enter number of columns  4
Enter temperature for top and sides  100
Enter temperature for bottom  50
Enter equilibrium tolerance  40
Initial Temperatures
   100   100   100   100
   100     0     0   100
   100     0     0   100
    50    50    50    50
Equilibrium Temperatures
  100.0000  100.0000  100.0000  100.0000
  100.0000   71.8750   71.8750  100.0000
  100.0000   59.3750   59.3750  100.0000
   50.0000   50.0000   50.0000   50.0000
```

ამ ამოცანის ამოხსნისას tolerance მნიშვნელოვანი ცვლადია. სცადეთ შეასრულოთ იგივე ამოცანა tolerance მცირე მნიშვნელობისათვის, მაგალითად 1 გრადუსი და დააკვირდით განსხვავებას შედეგად მიღებულ ტემპერატურული წონასწორობის სურათებს შორის. ასევე საინტერესო იქნება პროგრამა ისე შეცვალოთ, რომ დამატებით გვაძლევდეს იტერაციათა რაოდენობას, რომელიც აუცილებელია ტემპერატურული წონასწორობის დასამყარებლად.

ამ თავში განვიხილეთ პროგრამის კონტროლის სტრუქტურის 3 განსხვავებული ტიპის ოპერატორი. **if** ბრძანება, ოპერატორებთან **else** და **elseif** ერთად საშუალებას გვაძლევს გავუშვათ ბრძანებათა სხვადასხვა ჯგუფი, იმის მიხედვით, თუ როგორია ლოგიკური

გამოსახულება. **for** და **while** ბრძანებაები, რომელნიც ბრძანებთა ჯგუფის რამდენჯერმე განმეორების საშუალებას იძლევა. კონტროლის ეს სტრუქტურა აუცილებელია მრავალი საინჟინრო პრობლემის გადასაწყვეტად. განვიხილეთ ოპტიკურ ბოჭკოებთან და ტემპერატურულ წონასწორობასთან დაკავშირებული ამოცანები.

specsimboloebi

<	ნაკლებია
<=	ნაკლებია ან ტოლია
>	მეტია
>=	მეტია ან ტოლია
==	ტოლია
~=	არ უდრის
&	და
|	ან
~	არ

brZanebebi da funqciebi

all	განსაზღვრავს ჭეშმარიტია თუ არა ყველა მნიშვნელობა
any	განსაზღვრავს რომელიმე მნიშვნელობა მაინც თუ არის ჭეშმარიტი
break	შეწყვეტს პროგრამის მიერ გაშვებულ ციკლს
else	**if** სტრუქტურის ოპერატორი
elseif	**if** სტრუქტურის ოპერატორი
end	განსაზღვრავს კონტროლის სტრუქტურის დასასრულს
exist	განსაზღვრავს მოცემულ სამუშაო სივრცეში არის თუ არა მითითებული ცვლადი
find	პოულობს არანულოვან მნიშვნელობების ინდექსებს
finite	განსაზღვრავს სასრულია თუ არა სიდიდე
for	აწარმოებს ციკლის სტრუქტურას
if	ამოწმებს ლოგიკურ გამოსახულებას
isempty	განსაზღვრავს ცარიელია თუ არა მატრიცა
isnan	განსაზღვრავს განუზღვრელობაა თუ არა მოცემული სიდიდე
isstr	გაბსაზღვრავს ცვლადი სიმბოლოა თუ რიცხვითი გამოსახულება
strcmp	ადარებს ერთმანეთს ორ სტრიქონს
while	აყალიბებს ციკლის სტრუქტურას

ამოცანები

1 – 8 პერობლემა დაკავშირებულია ამ თავში განხილულ ამოცანებთან, 9 –23 შეეხება განსხვავებულ საინჟინრო პრობლემებს

ოპტიკური ბოჭკოები. ეს ამოცანები დაკავშირებულია გარდატეხის მაჩვენებელთა ცხრილთან, რომელიც მოცემული იყო ამ თავის შესაბამის ნაწილში.

1. დაბეჭდეთ ცხრილი, რომელიც შეიცავს ინფორმაციას ფაილიდან indices.dat შემდეგი ფორმატით:

refraction indices
Rod Index medium Index

2. დაუმატეთ 1 ამოცანაში აღწერილ ცხრილს სვეტი, რომელი შეიცავს კრიტიკულ კუთხეს გრადუსებში
3. დაუმატეთ 1 ამოცანაში აღწერილ ცხრილს სვეტი, რომელი შეიცავს კრიტიკულ კუთხეს რადიანებში
4. დაბეჭდეთ ინფორმაცია ფაილიდან indices.dat, რომელიც დააკავშირებს ისეთ მასალებს, რომლებიც შექმნიან სინათლის გამტარს. შემდეგი ფორმატით

Light Transmitting Pipes
Rod Index medium Index Angles of Incidense

5. შეცვალე 4 ამოცანით განსაზღვრული ცხრილი ისე, რომ კუთხე ნაცვლად გრადუსებისა, დაიბეჭდოს რადიანებში

თერმული წონასწორობა. ეს პრობლემები უკავშირდება ამ თავში განხილულ შესაბამის ამოცანას

6. ტემპერატურულ წონასწორობასთან დაკავშირებული პროგრამა შეცვალე ისე, რომ მეტალის ფირფიტის ოთხივე კიდეს განსხვავებული ტემპერატურეა ჰქონდეს
7. ტემპერატურულ წონასწორობასთან დაკავშირებული პროგრამა შეცვალე ისე, რომ დაბეჭდოს დამატებით იტერაციათა რაოდენობა, რომელიც საჭიროა თერმული წონასწორობის დასამყარებლად
8. ტემპერატურულ წონასწორობასთან დაკავშირებული პროგრამა შეცვალე ისე, რომ მეტალის ფირფიტის მხოლოდ ერთ კიდეს ჰქონდე ნულისაგან განსხვავებული ტემპერატურა. წერტილს შეიძლება ჰქონდეს ერთი, ორი ან სამი მეზობელი ტემპერატურა.

რაკეტის ტრაექტორია. ქარის პარამარტების გამოსაკვლევად შეიქმნა მცირე ზომის რაკეტა. ვიდრე ტესტირება დაიწყება, საჭიროა შესრულდეს რაკეტის ტრაექტორიის მოდელირება. ინჟინერთა გამოთვლით რაკეტის ტრაექტორია აღიწერება ფორმულით:

$$height = 60 + 2.13t^2 - 0.0013t^4 + 0.000034t^{4.751}$$

განტოლება გვაძლევს რაკეტის სიმაღლეს დედამიწის ზედაპირიდან დროის მოცემულ t მომენტში. სიმაღლის პირველი მნიშვნელობა (60) არის რაკეტის სიმაღლე, ანუ მანძილი დედამიწის ზედაპირიდან რაკეტის წვერომდე.

9. დაწერეთ პროგრამა, რომელიც გამოითვლის და დაბეჭდავს დროისა და რაკეტის სიმაღლის შესაბამისი მნიშნელობებს t = 0 დან იმ მომენტამდე, როცა რაკეტა დაეცემა დედამოწაზე. აიღეთ დროის ნაზრდი 2 წამი. თუ რაკეტა დედამიწაზე არ დაეცემა 100 წამის შემდეგ, შეწყვიტეთ პროგრამა.
10. 9 ამოცანაში აღწერილი პროგრამა შეცვალეთ ისე, რომ ცხრილის ნაცვლად დაბეჭდოს დროის მნიშვნელობა, როცა რაკეტა დედამიწაზე დაშვებას იწყებს და როცა იგი შეეხება დედამიწას.

საოპერაციო ძაფის დაფასოება. მედიცინაში ცოცხალი ქსოვილის გასაკერად ოპერაციის შემდეგ გამოიყენება სპეციალური ბოჭკო. მათი დაფასოებისას დიდი სიფრთხილეა საჭირო, რომ მტვერი და მიკრობები არ მოხვდეს პაკეტში. შეფუთვის შემდეგ ხდება პაკეტის დაბეჭდვა. შტამპი, რომელიც ლუქავს პაკეტს ცხელდება ელექტროგამათბობლით. იმისათვის, რომ დალუქვის პროცესი დამაკმაყოფილებლად ჩაითვალოს, შტამპმა უნდა შეინარჩუნოს საჭირო დონეზე განსაზღვრული ტემპერატურა, წნევა, რმლითაც ის პაკეტს აწვება და დროის ინტერვალი ორ მომდევნო ოპერაციას შორის. დროის პერიოდს ორ ურთირთმომდევნო კონტაქტს შორის (dwell time)- შეყოვნება ეწოდება. დავუშვათ დასაშვები მნიშვნელობების ინტერვალი დამაკმაყოფილებელი პროცესისათვის ასეთია:

ტემპერატურა	150-170^{0}C
წნევა	60-70 psi
შეყოვნება	2-2.5 წმ

11. მონაცემთა ფაილი suture.dat შეიცავს ინფორმაციას ერთი კვირის განმავლობაში წუნდებული სამედიცინო ძაფის პარტიის შესახებ. ყოველი სტრიქონი ფაილში შეიცავს პარტიის ნომერს, ტემპერატურას, წნევას, შეყოვნების პერიოდს დაწუნებული პარტიისათვის. ხარისხის მაკონტროლებელმა ინჟინერმა უნდა გააანალიზოს ეს ინფორმაცია, რათა შეაფასოს რამდენი პროცენტია წუნდებულ პაკეტებში გამოწვეული ტემპეტატურის, წნევის და შეყოვნების დეფექტის გამო. შესაძლოა ზოგიერთი პარტია წუნდებულია ორი სხვადასხვა ნიშნით, მაშინ იგი რეგისტრირდება ყველა შემთხვევაში ცალკ-ცალკე. დაწერეთ პროგრამა, რომელიც დაითვლის და დაბეჭდავს სხვადასხვა ტიპის წუნის პროცენტულ შედგენილობას.
12. 11 ამოცანაში აღწერილი პროგრამა შეცვალეთ ისე, რომ იგი აგრეთვე ბეჭდავდეს პარტიათა ნომერს წუნის თითოეული კატეგორიისათვის და პარტიათა საერთო რაოდენობას, რომელიც ამოღებული იქნა. ყურადღება მიაქციეთ, დაწუნებული პარტია საერთო რაოდენობაში მხოლოდ ერთხელ უნდა ფიგურირებდეს, თუმცა შესაძლოა რამდენჯერმე შეგვხვდეს სხვადასხვა კატეგორიით დაწუნებულ პარტიებში.
13. დაწერეთ პროგრამა, რომელიც წაიკითხავს ფაილს suture.dat, დარწმუნდით, რომ მონაცემებში მართლაც წუნდებული პარტიები მოყვა. თუ რომელიმე პარტია შეცდომითაა მოხვედრილი ფაილში, პროგრამამ დაგვიბეჭდოს სათანადო ცნობა ამის თაობაზე.

ტყის განახლება. ამოცანა ეხება პრობლემას ხეტყის დამუშავებაში. ტყის ფართობის რა ნაწილი უნდა დარჩეს გაუჩეხავი, რომ მოხდეს ტყის განახლება სასურველ პერიოდში. ითვლება, რომ გაჩეხილი ტყის განახლებას გარკვეული დრო სჭირდება ნიადაგისა და კლიმატის პირობების გათვალისწინებით. ტყის განახლების განტოლება გამოხატავს ამ დროს როგორც გაუჩეხავად დარჩენილი ტყის ფართობის და განახლების კოეფიციენტის (სიჩქარის) ფუნქციას. მაგალითად, ვთქვათ ხეტყის დამზადების შემდეგ 100 აკრი ტყე დარჩა გაუჩეხავი, ხოლო ტყის განახლების კოეფიციენტია 0.05, მაშინ პირველი წლის ბოლოს გვექნება 100+0.5 * 100, ანუ 105 აკრი განახლებული ტყე, მეორე წლის ბოლოს კი – 105 + 0.05 * 105=110.25 აკრი იქნება განახლებული ტყის საერთო ფართი.

14. დავუშვათ გვაქვს ტყე საერთო ფართით 14000 აკრი, სადაც დარჩენილი გაუჩეხავი ფართი 2500 აკრია და ტყის განახლების კოეფიციენტია 0.02. დაბეჭდეთ ცხრილი, რომელიც უჩვენებს განახლებული ტყის ფართს ყოველი წლის ბოლოს 20 წლის განმავლობაში.
15. 14 ამოცანაში აღწერილი პროგრამა შეცვალეთ ისე, რომ საშუალება გვქონდეს შევიყვანოთ მონაცემი წლების რაოდენობის შესახებ.

16. 14 ამოცანაში აღწერილი პროგრამა შეცვალეთ ისე, რომ საშუალება გვქონდეს შევიყვანოთ მონაცემი აღდგენილი ტყის ფართის შესახებ და პროგრამამ განსაზღვროს რამდენი წელი დასჭირდება აასეთი ფართის აღდგენას.

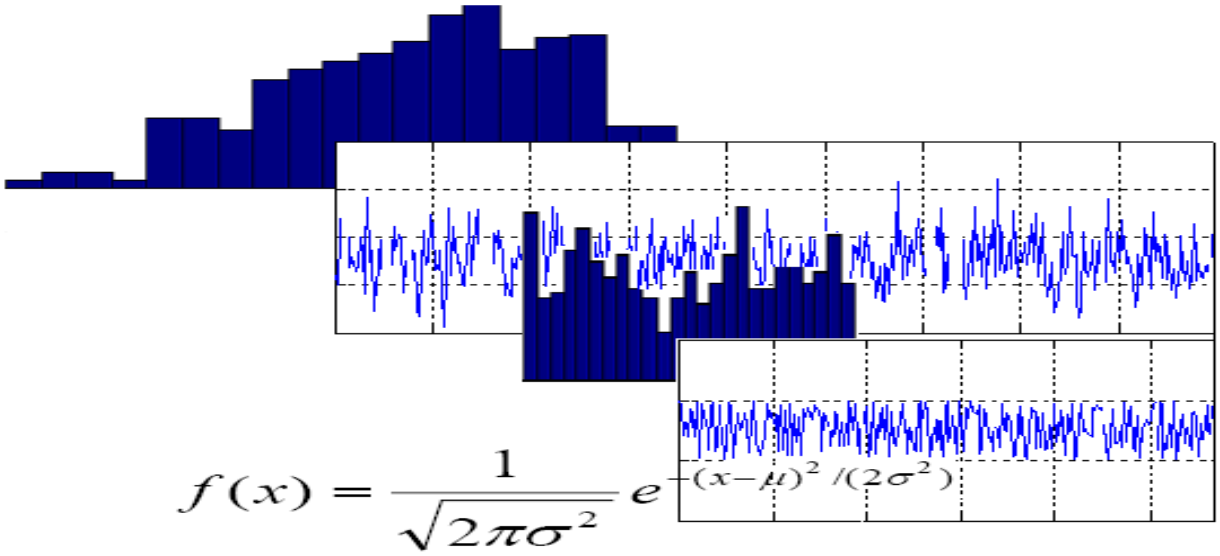
კრიტიკული ანალიზი. ეს არის ტექნიკა, რომელიც გამოიყენება პროექტის გეგმის განრიგის შესადგენად. ეს ინფორმაცია სასარგებლოა პროექტის შესრულებისათვის თვალყურის სადევნებლად. ასეთი ანალიზის ერთ-ერთი მეთოდი შემდეგში მდგომარეობს: დავყოთ პროექტი თანმიმდევრულ მოვლენებად. ყოველი მოვლენა დავყოთ რამდენიმე ქვეპუნქტად. ისე, რომ ერთი მოვლენა უნდა დასრულდეს, ვიდრე მეორე დაიწყება. მოვლენის შიგნით რამდენიმე ქვეპუნქტი შესაძლოა ერთდროულად დაიწყოს და მიმდინარეობდეს. დრო, რომელიც საჭიროა ერთი მოვლენის დასამთავრებლად დამოკიდებულია დღეების რაოდენობაზე, რომელიც საჭიროა მისი ყველაზე ხანგრძლივი ქვეპუნქტის შესასრულებლად. მსგავსად ამისა პროექტის დასრულების დრო ტოლია მოვლენათა დასრულებისათვის საჭირო დროის ჯამისა. დავუშვათ კრიტიკული ანალიზის ინფორმაცია პროექტისათვის ჩაწერილია ფაილში: path. dat. ამ ფაილის ყოველი სტრიქონი შეიცავს მოვლენის ნომერს, ქვეპუნქტის ნომერს და ამ ქვეპუნქტის შესრულებისათვის სჭირო დღეების რაოდენობას. მონაცემები ისეა დალაგებული, რომ ჯერ მოცემულია პირველი მოვლენის ყველა ქვეპუნქტი. ამას მოჰყვება მეორე მოვლენის ყველა ქვეპუნქტი და ა.შ.

17. დაწერეთ პროგრამა, რომელიც წაიკითხავს ინფორმაციას ფაილიდან და დაბეჭდავს პროექტის დასრულების განრიგს. ცხრილში მოცემული უნდა იყოს თითოეული მოვლენის ნომერი, ყველაზე ხანგრძლივი ქვეპუნქტის დასრულებისათვის საჭირო დღების რაოდენობა, პროექტის დასრულებისათვის საჭირო დრო დღეებში.
18. წინა ამოცანაში აღწერილი პროგრამა შეცვალე ისე, რომ პროგრამამ დაბეჭდოს მხოლოდ ის მოვლენა და ქვეპუნქტი, რომლის დასრულებისათვის საჭიროა 5 დღეზე მეტი.
19. შეცვალე პროგრამა ისე, რომ დაბეჭდოს მოვლენის ნომერი და მასში შემავალი ქვეპუნქტების რაოდენობა.

მონაცემები მიმღებიდან. დავუშვათ ფაილი სახელით **sensor.dat** შეიცავს ინფორმაციას, რომელიც შეგროვილია რამდენიმე სენსორიდან ერთდროულად. ყოველი სტრიქონი შეიცავს სხვადასხვა სენსორიდან აღებულ მონაცემებს. პირველ სტრიქონში გვაქვს მონაცემები აღებული t = 0.0 წმ-ზე, მეორეში - 1.0 წმ-ზე და ა. შ.

20. დაწერეთ პროგრამა, რომელიც კითხულობს მონაცემთა ფაილს და ბეჭდავს სენსორების რაოდენობას და და ინფორმაციას იმის შესახებ, რამდენი წამის განმავლობაში გროვდებოდა მონაცემები.
21. შეცვალე მე-20 ამოცანისათვის დაწერილი პროგრამა ისე, რომ ყველა მონაცემი, რომელიც აჭარბებს 10.0 გაუტოლდეს 10.0, ხოლო ყველა მონაცემი, რომელიც ნაკლებია –10, გაუტოლდეს –10.0
22. დაწერეთ პროგრამა, რომელიც ააგებს პირველ მიმღებზე შეგროვილი მონაცემების გრაფიკს დროის მიმართ.
23. დაწერეთ პროგრამა, რომელიც იპოვის ცხრილში იმ მონაცემის მდებარეობას (ინდექსს), რომლის მნიშვნელობაც სასრული სიდიდე არ არის
24. დაწერეთ პროგრამა, რომელიც განსაზღვრავს იმ მონაცემთა მდებარეობას, რომელთა მნიშვნელობაც მეტია 20-ზე
25. დაწერეთ პროგრამა, რომელიც დაბეჭდავს ისეთ მონაცემთა რაოდენობას, რომელთა მნიშვნელობაც 0-ის ტოლია.

-



5 სტატისტიკური გაზომვები

პრობლემა: ზეგამტარობა
შესავალი
5.1 ფუნქციები მონაცემთა ანალიზისათვის
პრობლემა: საკომუნიკაციო სიგნალის ანალიზი
5.1 შემთხვევითი სიდიდე
პრობლემა: თვითმფრინავის ფრენის მოდელირება
5.1 თანაფარდობა სიგნალი - ხმაური
პრობლემა: ტემპერატურული წონასწორობა
დასკვნა

შესავალი

ექსპერიმენტის ჩატარებისას შეგროვილ მონაცემთა ანალიზი მნიშვნელოვანი პუნქტია საინჟინრო ექსპერიმენტის შესაფასებლად. ანალიზი მოიცავს როგორც მარტივ გამოთვლებს აგრეთვე მონაცემთა საშუალო მნიშვნელობის, სტანდარტული გადახრისა და დისპერსიის გამოთვლას. ეს სიდიდეები სტატისტიკური სიდიდეებია, რადგან მათ აქვთ სტატისტიკური თვისებები. მაგალითად $\sin 60^0$ ზუსტი რიცხვია. ის არ შეიცვლება, რამდენჯერაც არ უნდა გამოვითვალოთ, მაგრამ მანძილი, რომელსაც მანქანა გაივლის 1 გალონი საწვავის გამოყენებით, სტატისტიკური სიდიდეა, რადგან იგი გამოკიდებულია სხვადასხვა პარამეტრებზე, როგორიცაა ტემპერატურა, სიჩქარე, რომელსაც მანქანა ავითარებს, გზის ტიპი, რომელზეც მოძრაობს. შეგვიძლია შევაფასოთ სტატისტიკური მონაცემების მახასიათებლები, ასევე შეგვიძლია კომპიუტერის საშუალებით ვაწარმოოთ რიცხვითი მწკრივები (შემთხვევითი სიდიდეები) განსაზღვრული მახასიათებლებით. ამ თავში ვნახავთ, როგორ გამოვიყენოთ მონაცემთა ანალიზის ფუნქციები MATLAB–ში და როგორ შევქმნათ შემთხვევით სიდიდეთა განაწილება, რომელსაც ექნება განსაზღვრული სტატისტიკური მახასიათებლები.

1.1 ფუნქციები მონაცემთა ანალიზისათვის

MATLAB შეიცავს ფუცქციათა რიგს მონაცემთა სტატისტიკური ანალიზისათვის. ჯერ განვიხილოთ რამდენიმე მარტივი ფუნქცია.

ფუნქციათა შემდეგი ჯგუფი ხშირად გამოიყენება მონაცემთა ანალიზისათვის.

4.1.1 მაქსიმუმი და მინიმუმი

მონაცემთა ისეთი პარამეტრების შესაფაებლად, როგორიცაა მაქსიმუმი და მინიმუმი, გვაქვს ფუნქციები:

max (x)	თუ x ვექტორია, ეს ფუნქცია გვაძლევს მის ელემენტებს შორის უდიდესის მნიშვნელობას, თუ მატრიცაა, მაშინ – სტრიქონ – ვექტორს, რომლის ელემენტებიც თითოეული სვეტის ელემენტთა შორის უდიდესის ტოლია
max (x,y)	გვაძლევს იგივე ზომის მატრიცას, როგორიცაა x და y, რომლის ყოველი ელემენტი x და y შესაბამის ელემენტებს შორის უდიდესის ტოლია
[y,I] = max (x)	ასეთი ფორმით ბრძანება აგბს y ვექტორს x მატრიცის ყოველი სვეტის მაქსიმუმის მნიშვნელობებით, ხოლო ამ ელემენტების შესაბამის ინდექსებს გვაძლევს i ვექტორის სახით. თუ ორო მაქსიმუმი ერთმანეთს დაემთხვა, პირველი მნიშვნელობის ინდექსს გვაძლევს
min (x)	თუ x ვექტორია, ეს ფუნქცია გვაძლევს x უმცირეს მნიშვნელობას, თუ x მატრიცაა მაშინ – სტრიქონ – ვექტორს, რომლის ელემენტებიც თითოეული სვეტის ელემენტთა შორის უმცირესის ტოლია
min (x, y)	გვაძლევს იგივე ზომის მატრიცას, როგორიცაა x და y, რომლის ყოველი ელემენტი x და y შესაბამის ელემენტებს შორის უმცირესის ტოლია
[y, I] = min (x)	ასეთი ფორმით ბრძანება აგბს y ვექტორს x მატრიცის ყოველი სვეტის მინიმუმის მნიშვნელობებით, ხოლო ამ ელემენტების შესაბამის ინდექსებს გვაძლევს i ვექტორის სახით. თუ ორო მინიმუმი ერთმანეთს დაემთხვა, პირველი მნიშვნელობის ინდექსს გვაძლევს

საშუალო და მედიანა. რიცხვით სიდიდეთა ჯგუფის საშუალო მათი არითმეტიკული საშუალოა. საშუალოს აღნიშნავენ ბერძნული სიმბოლოთი μ(მიუ)

$$\mu = \frac{\sum_{k=1}^{N} x_k}{N}$$

სადაც

$$\sum_{k=1}^{N} x_k = x_1 + x_2 + x_3 + \cdots + x_n$$

მედიანა არის ჯგუფის საშუალედო მნიშვნელობა, თუ ჯგფის წევრებს დავალაგებთ ზრდის მიხედვით. თუ x ვექრორი შეიცავს ეელემენტთა კენტ რაოდენობას N, ჯერ დავალაგებთ x ელემენტებს ზრდის მიხედვით, მედიანა იქნება იმ ელემენტის ტოლი, რომლის ინდექსია **ceil**(N/2) (თუ N=5, მედიანა იქნება მესამე ელემენტის ტოლი), თუ N ლუწია, მაშინ მედიანა იქნება (სრდის მიხედვით დალაგების შემდეგ) შუა ორი ელემენტის საშუალო მნიშვნელობა.

MATLAB საშუალოს და მედიანას გამოითვლის ფუნქციით:

mean (x)	თუ x ვექტორია, ეს ფუნქცია გამოითვლის x ელემენტების საშუალო მნიშვნელობას. თუ x მატრიცაა, ეს ფუნქცია გვაძლევს ვექტორს, რომლის ელემენტებია შესაბამისი სვეტების ელემენტთა საშუალო მნიშვნელობა
median (x)	თუ x ვექტორია, ეს ფუნქცია გამოითვლის x ელემენტების მედიანას. თუ x მატრიცაა, ეს ფუნქცია გვაძლევს ვექტორს, რომლის ელემენტებია შესაბამისი სვეტების ელემენტთა მედიანა

4.1.2 ჯამი და ნამრავლი

MATLAB აქვს ფუნქციები, რომლებიც გამოითვლის მატრიცის ელემენტების ჯამს და ნამრავლს და აგრეთვე ისეთი ფუნქციები, რომებიც იძლევა მატრიცის ელემენტების კუმულაციურ ჯამსა და ნამრავლს

sum (x)	თუ x ვექტორია, ეს ფუნქცია გამოითვლის x ელემენტების ჯამს. თუ x მატრიცაა, ეს ფუნქცია გვაძლევს ვექტორს, რომლის ელემენტებია შესაბამისი სვეტების ელემენტთა ჯამს
prod (x)	თუ x ვექტორია, ეს ფუნქცია გამოითვლის x ელემენტების ნამრავლს. თუ x მატრიცაა, ეს ფუნქცია გვაძლევს ვექტორს, რომლის ელემენტებია შესაბამისი სვეტების ელემენტთა ნამრავლი
cumsum (x)	თუ x ვექტორია, ეს ფუნქცია გვაძლევს იგივე სიგრძის ვექტორს, რომლის ელემენტები x ელემენტების კუმულაციურ ჯამს წარმოადგენ, თუ x მატრიცაა, ეს ფუნქცია იძლევა იგივე ზომის მატრიცას, რომლის ელემენტებიც კუმულაციურ ჯამს წაროადგენს სვეტების მიმართ
cumprod (x)	თუ x ვექტორია, ეს ფუნქცია გვაძლევს იგივე სიგრძის ვექტორს, რომლის ელემენტები x ელემენტების კუმულაციურ ნამრავლს წარმოადგენ, თუ x მატრიცაა, ეს ფუნქცია იძლევა იგივე ზომის მატრიცას, რომლის ელემენტებიც კუმულაციურ ნამრავლს წაროადგენს სვეტების მიმართ

sort (x)	თუ x ვექტორია, დაალაგებს ელემენტებს ზრდის მიხედვით. თუ x მატრიცაა, ეს ფუნქცია თითოეულ სვეტს დაალაგებს ზრდის მიხედვით
[y, i]=sort(x)	ასეთი ფორმით ეს ფუნქცია y ვექტორის სახით მოგვცემს x ვექტორის მნშვნელობებს დალაგებულს ზრდის მიხდვით ხოლო i ვექტორში მოგვცემს შესაბამისი ინდექსების მნიშვნელობას.

სავარჯიშო

მოცემულია მატრიცები:

$$w = [\ 0\ 3\ -2\ 7];$$
$$x = [3\ -1\ 5\ 7];$$

$$y = \begin{bmatrix} 1 & 3 & 7 \\ 2 & 8 & 4 \\ 6 & -1 & -2 \end{bmatrix}$$

გამოითვალეთ შემდეგი სიდიდეები და შეამოწმეთ MATLAB საშუალებით

1. max (w)
2. min (y)
3. min (w,x)
4. [z, i] = max (y)
5. mean (y)
6. median (w)
7. cumprod (y)
8. sum (x)
9. sort (2*w + x)
10. sort (y)

4.1.3 დისპერსია და სტანდარტული გადახრა

მონაცემთა ჯგუფისათვის ერთ-ერთი მნიშვნელოვანი სტატისტიკური მახასიათებელია დისპერსია. ვიდრე განვიხილავდეთ დისპერსიის მათემათიკურ შინაარსს, განვიხილოთ ასეთი მაგალითი: გვაქვს მონაცემთა ორი ჯგუფს data1 და data2 (ნახ. 5.1) ნახაზი აგებულია subplot ბრძანებით, რომელსაც მოგვიანებით გავეცნობით. თუ შევეცდებით გავატაროთ მრუდი საშუალო მნიშვნელობებზე, ეს მრუდი ორივე მათგანისთვის შეესაბამება მნიშვნელობას 3.0. ამიტომ შეგვიძლია ვივარაუდოთ, რომ ორივე ჯგუფის საშუალო მნიშვნელობა 3.0 ტოლია, თუმცა მონაცემებს განსხვავებული მახასიათებლები გააჩნიათ. data2 მონაცემები უფრო მკვეთრად გადაიხრება საშუალო მნიშვნელობიდან. ე. ი. ცვლილების, გადახრის

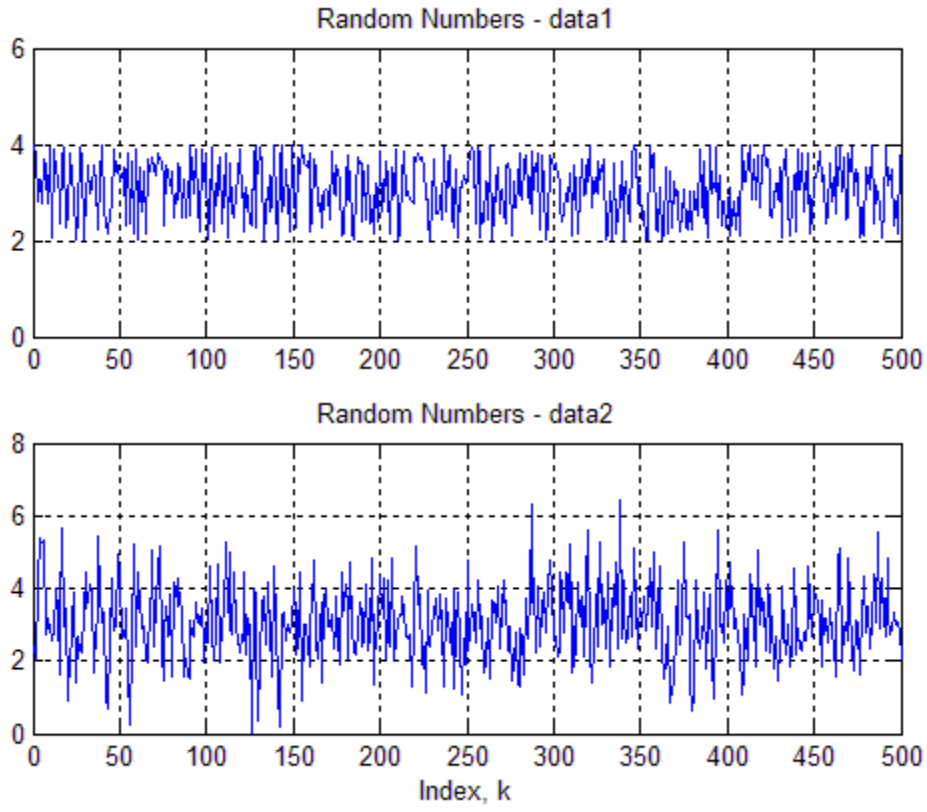
მნიშვნელობები data2 უფრო დიდია, ვიდრე data1 –თვის. რაც უფრო დიდია გადახრა, უფრო მეტად ფლუქტუირებენ მონაცემები საშუალო მნიშვნელობის მიმართ.

მათემატიკურად დისპერსიას აღნიშნავენ ბერნული სიმბოლოთი σ^2(სიგმა), მონაცემთა მწკრივის (რომელიც x ვექტორის ელემენტებს წარმოადგენს) დისპერსია გამოითვლება შემდეგი ფორმულით:

$$\sigma^2 = \frac{\sum_{k=1}^{N}(x_k - \mu)^2}{N-1}$$

თუ კარგად დააკვირდებით გამოსახულება საკმაოდ მარტივია. x_k - μ წარმოადგენს ვექტორის k ური ელემენტის გადახრას საშუალოდან, ეს მნიშვნელობა კვადრატშია აყვანილი, რომ ჯამის ყველა შესაკრები დადებითი სიდიდე იყოს, შემდეგ აღებულია ჯამი საშუალოდან თითოეული ელემენტის გადახრისა. საბოლოოდ მიღებული ჯამი გაყოფილი N-1 – ელემენტთა რაოდენობაზე. ეს განტოლება გამოითვლის დისპერსიას. კვადრატული ფესვის მნიშვნელობას დისპერსიიდან, უწოდებენ საშუალო კვადრატულ ან სტანდარტულ გადახრას:

$$\sigma = \sqrt{\sigma^2}$$



MATLAB აქვს ფუნქცია სტანდარტული გადახრის გამოსათვლელად. თუ დისპერსია გვაინტერესებს, უბრალოდ, სტანდარტულ გადახრას კვადრატში ავიყვანთ.

std (x) თუ x ვექტორია, ეს ფუნქცია გამოითვლის x მნიშვნელობათა სტანდარტულ გადახრას. თუ x მატრიცაა, მოგვცემს

სტრიქონ ვექტორს, რომლის ელემენტებია შესაბამისი სვეტის ელემენტთა სტანდარტული გადახრაა.

სავარჯიშო

განსაზღვრეთ შემდეგ ფუნფციათა მნიშვნელობები, თუ მოცემულია მატრიცები:

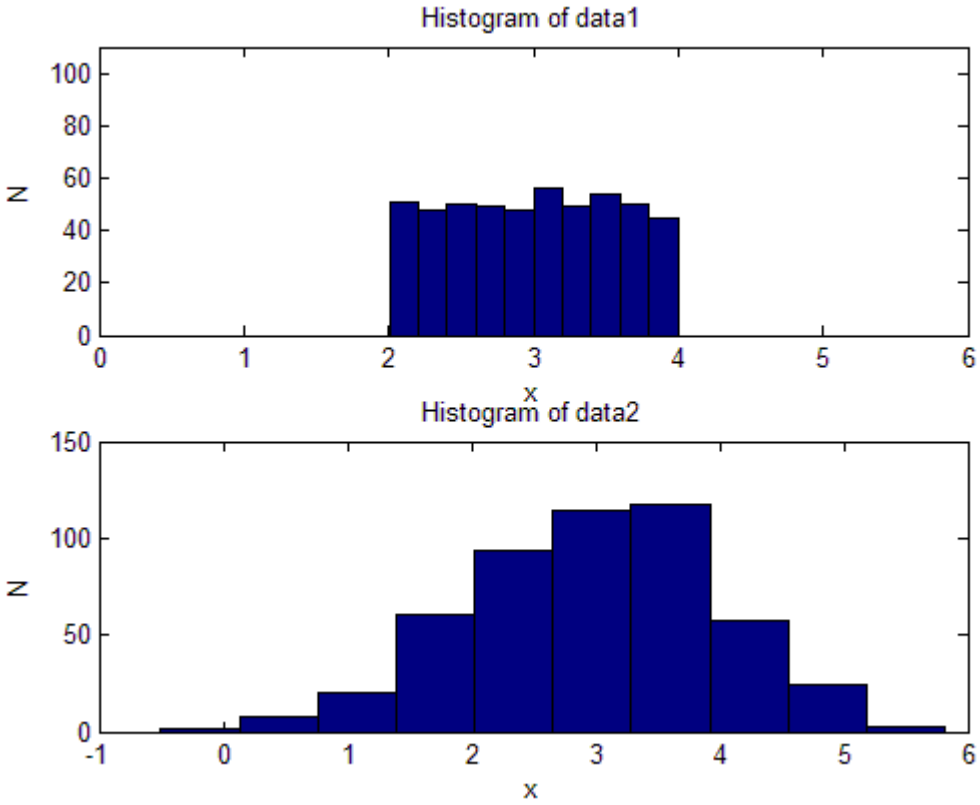
w = [0 3 –2 7];
x = [3 –1 5 7];

$$y = \begin{bmatrix} 1 & 3 & 7 \\ 2 & 8 & 4 \\ 6 & -1 & -2 \end{bmatrix}$$

1. std (w)
2. std (x)^2
3. std (y(:2))
4. std (y)
5. std (y).^2

4.1.4 ჰისტოგრამა

MATLAB –ის გრაფიკული შესაძლებლობები განხილული იქნება მე-7 თავში. აქ განვიხილავთ ჰისტოგრამას, რომელიც სპეციალური გრაფიკული საშუალებაა სტატისტიკური მონაცემების ვიზუალიზციისათვის. ჰისტოგრამა გვიჩვენებს მონაცემთა განაწილების სურათს. MATLAB –ში ჰისტოგრამა გამოითვლის იმ სიდიდეთა რაოდენობას, რომელნიც ხვდებიან 10 ზოლში, როლებადაც დაყოფილია მონაცემთა უმცირეს და უდიდეს მნიშვნელობას შორის ინტერვალი. მაგალოთად, თუ ავაგებთ data1 და data2 მონაცემების ჰისტოგრამას, მივიღებთ ნახ, 5.2.



ნახ. 5.2 ჰისტოგრამა 10 სვეტით

შევნიშნავთ, რომ ჰისტოგრამა განსხვავებულ ინფორმაციას გვაძლევს. ჰისტოგრამა გვაძლევს არა მარტო მნიშვნელობათა არეს, არამედ ინფორმაციას იმის თაობაზე, თუ როგორ არის ეს მონაცემები განაწილებული. მაგალითად data1 მონაცემები უჩვენებს რომ მონაცემები თითქმის თანაბრადაა განაწილებული უმცირეს (2) და უდიდეს (4) სიდიდეს შორის (ასეთი ტიპის განაწილებას თანაბარი ეწოდება). data2 მონაცემები არ არის განაწილებული თანაბრად, მონაცემთა უმრავლესობა კონცენტრირებულია საშუალო მნიშვნელობის მახლობლად (ასეთი ტიპის განაწილებას გაუსისებური, ან ნორმალური განაწილება ეწოდება).

MATLAB ბრძანება ჰისტოგრამის ასაგებად ამგვარია:

hist (x)

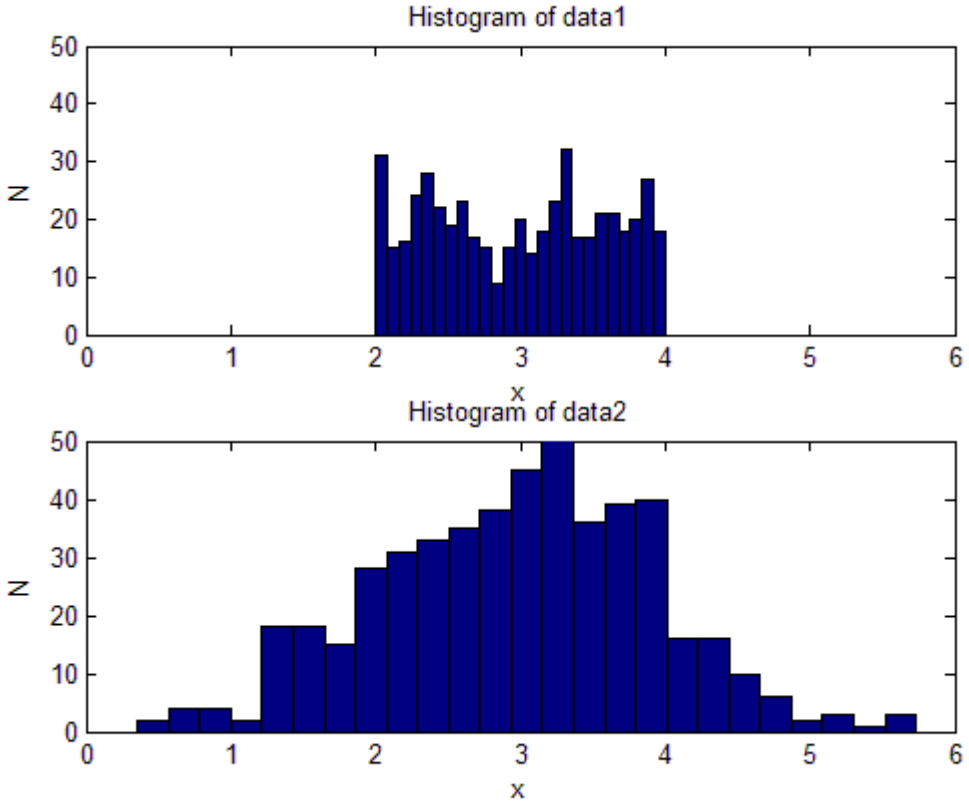
სადაც x ვექტორია იმ მონაცემებით, რომლის ჰისტოგრამაც უნდა ავაგოთ. ეს ბრძანება პირველ რიგში დაალაგებს x ვექტორის ელემენტებს ზრდის მიხედვით, შემდეგ ინტერვალს x-ის მინიმალურ და მაქსიმალურ მნიშვნელობებს შორის გაყოფს 10 ტოლ ნაწილად და დაითვლის თითოეულ სექტორში რამდენი მონაცემი მოხვდა.

თუ მოცემული გვაქვს მონაცემები მატრიცის სახით და გვსურს ავაგოთ მისი მეორე სვეტის მნიშვნელობათა ჰისტოგრამა, ასე უნდა მივუთითოთ:

hist (data (: , 2))

ბრძანება **hist** ასევე საშუალებას იძლევა შევარჩიოთ ჰისტოგრამის სვეტების რაოდენობა. თუ გვსურს გავზარდოთ ჰისტოგრამის გარჩევა ისე, რომ იგი შეიცავდეს 25 სვეტს ნაცვლად ათისა, მივმართავთ შემდეგ ბრძანებას:

hist (x, 25)



ნახ. 5.3 ჰისტოგრამა 25 სვეტით

შესაბამისი ჰისტოგრამა ნაჩვენებია ნახ. 5.3. (5.2 და 5.3 ნახაზები აგებულია MATLAB ბრძანებებით **hist** და **subplot,** რომელთაც მოგვიანებით შევისწავლით).
ინფორმაცია, რომლის მიხედვითაც ჰისტოგრამა აიგება, შეგვიძლია ვექტორის სახითაც ჩავწეროთ:

[n, x] = hist (data1);
[n, x] = hist (data1,25);

ეს ბრძანებები ჰისტოგრამას კი არ ააგებენ, არამედ შექმნიან ორ ვექტორს – n და x. n ვექტორი შეიცავს დათვლებს (თითოეულ სვეტში მოხვედრილი მონაცემების რაოდენობა) 10 სვეტისათვის, x ვექტორი კი - თითოეული სვეტის შუაწერტილის მნიშვნელობას. მეორე ბრძანება იგივეა, რაც პირველი, მაგრამ 25 სვეტისათვის.
ეს ვექტორები დაგვჭირდება **bar** გრაფიკების ასაგებად, რომელსაც მე-7 თავში განვიხილავთ.

პრობლემა: საკომუნიკაციო სიგნალი

დავუშვათ გვინდა შევქმნათ სისტემა, რომელიც გამოიცნობს წარმოთქმულ სიტყვას შემდეგი ათი სიტყვიდან: "ნული", "ერთი", "ორი", "სამი". . . "ცხრა", რომლებიც ციფრული სიგნალის სახითაა წარმოდგენილი. პირველი, რაც უნდა გავაკეთოთ არის გავაანალოზოთ შესაბამისი სიგნალის მნიშვნელობები. ვნახით, არსებობს თუ არა რაიმე სტატისტიკური შეფასება, რომელიც მათ გამორჩევაში დაგვეხმარება. მონაცემთა ანალიზის MATLAB ფუნქციები მარტივად გამოითვლის სტატისტიკურ მახასიათებლებს,

შემდეგ შეგვიძლია დავბეჭდოთ მათი მნიშვნელობები, მოვძებნოთ ის პარამეტრები, რომლებიც დასმული ამოცანის გადაწყვეტაში დაგვეხმარება. მაგალითად შესაძლოა ერთი შედეგის საშუალებით 10 დან 3 მონაცემი გამოვარჩიოთ, შემდეგ კი მეორე შეფასების საშუალებით ამ სამს შორის ჩვენთვის საჭირო სიგნალი გამოვარჩიოთ.

ვისარგებლოთ რეალური სიგნალით, გამოვთვალოთ სტატისტიკური მახასიათებლები, რომლებიც შემდგომში გამოიყენება სიგნალის გამოცნობის სრულ ალგორითმში. თითოეული სიგნალის შესაბამისი მონაცემთა ფაილი შეიცავს 1000 სიდიდეს. დაწერეთ MATLAB პროგრამა, რომელიც წაიკითხავს ASCII მონაცემთა ფაილს "nuli.dat" და გამოითვლის შემდეგ ინფორმაციას: საშუალო, სტანდარტული გადახრა, დისპერსია, საშუალო სიმძლავრე, საშუალო ამპლიტუდა და ნულოვანი გადაკვეთა.

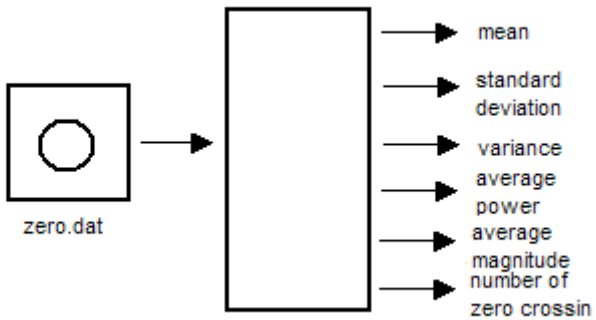
ზემოთ უკვე განვიხილეთ საშუალო, სტანდარტული გადახრა და დისპერსია. საშუალო სიმძლავრე არის საშუალო კვადრატული სიდიდე და უფრო დაწვრილებით შემდეგში იქნება განხილული. ნულოვანი გადაკვეთა გვიჩვენებს თუ მონაცემთა მწკრივში რამდენჯერ შეიცვალა უარყოფითი სიდიდე დადებითით და პირიქით.

1. ამოცანის დასმა

გამოითვალეთ საკომუნიკაციო სიგნალის სტატისტიკური მახასიათებლები: საშუალო, სტანდარტული გადახრა, დისპერსია, საშუალო სიმძლავრე, საშუალო ამპლიტუდა და ნულოვანი გადაკვეთის წერტილების რაოდენობა.

2. INPUT/OUTPUT აღწერა

ნახ 5.4 წარმოადგენს დიაგრამას. საწყისი მნიშვნელობა ეს არის ფაილი ხმოვანი სიგნალით, საბოლოო მნიშვნელობა კი ფაილის მონაცემთა სტატიტიკური მახასიათებლები.



ნახ. 5.4 INPUT/OUTPUT დიაგრამა

3. სახელდახელო ამოხსნა

დავუშვათ ფაილი შეიცავს შემდეგ მონაცემებს:

[2.5 8.2 –1.1 –0.2 1.5]

კალკულატორის საშუალებით შეგვიძლია გამოვითვალოთ:

საშუალო = (2.5 + 8.2 + –1.1 + –0.2 + 1.5)/5 = 2.18
დისპერსია = $[(2.5 - \mu)^2 + (8.2 - \mu)^2 + (-1.1 - \mu)^2 + (-0.2 - \mu)^2 + (1.5 - \mu)^2]/4$
=13.307

სტანდარტული გადახრა = $\sqrt{13.307}$ = 3.648

საშუალო სიმძლავრე = $(2.5^2 + 8.2^2 + (-1.1)^2 + (-0.2)^2 + 1.5^2)/5$
საშუალო ამპლიტუდა = (|2.5| + |8.2| + |-1.1| + |-0.2| + |1.5|)/5 = 2.7
ნულოვანი გადაკვეთის წერტილები = 2

4. MATLAB ამოხსნა

```
%
%     This program komputes a number of statistics
%     for an utterance stored in a data file
%
load one.dat;
x = one;
fprintf ('Digit Statistics \n\n')
fprintf ('mean: %f \n', mean (x))
fprintf ('standard deviation: %f \n', std(x))
fprintf ('variance: %f \n', std(x)^2)
fprintf ('average power: %f \n', mean(x.^2))
fprintf ('average magnitude: %f \n', mean(abs(x)))
crossings = 0;
for j = 1:length(x)-1
    if x(j)*x(j+1) < 0
        crossings =crossings + 1;
    end
end
fprintf ('zero crossings: %f \n', crossings)
```

5. შემოწმება

პროგრამა გავუშვით MATLAB–ში და მონაცემთა ფაილისათვის “one,dat” გამოვითვალეთ შემდეგი სტატისტიკური მნიშნელობები:

```
Digit Statistics

mean: 0.000494
standard deviation: 0.218396
variance: 0.047697
average power: 0.047696
average magnitude: 0.099959
zero crossings: 424.000000
```

1.2 შემთხვევითი რიცხვები

შემთხვევითი რიცხვები არ მოიცემა განტოლებით. მათი მნიშვნელობა შემთხვევაზეა დამოკიდებული. წინასწარ შეუძლებელია იმის განსაზღვრა, თუ რა მნიშვნელობას მიიღებს შემთხვევითი სიდიდე. ის რიცხვითი მნიშვნელობა, რომელიც შეუძლია მიიღოს შემთხვევიღ სიდიდეს, არის მისი შესაძლო მნიშვნელობა. შემთხვევითი სიდიდის შესასწავლად პირველ რიგში უნდა ვიცოდეთ ის რიცხვითი მნიშვნელობები, რომელთა მიღებაც მას შეუძლია. მაგრამ მარტო შესაძლო რიცხვით მნიშვნელობათა ცოდნა არ არის საკმარისი შემთხვევითი სიდიდის შესასწავლად. საჭიროა აგრეთვე ვიცოდეთ, თუ რა სიხშირით ღებულობს იგი ამა თუ იმ შესაძლო მნიშვნელობას. ამას კი განსაზღვრავს მისი განაწილების კანონი, რომელიც მოიცემა განაწილების ფუნქციით. მრავალ საინჟინრო ამოცანაში გვჭირდება შემთხვევითი სიდიდეების გამოყენება. ზოგ შემთხვევაში ისინი მოდელირებისთვისაა საჭირო. მოდელი შეაძლოა რამდენჯერმე გამოიცადოს და მოხდეს მიღებული შედეგის ანალიზი, ყოველი ხელახალი ტესტირება შეესაბამება ექსპერიმენტის განმეორებას შეცვლილი პარამეტრებით. შემთხვევითი რიცხვები გვჭირდება ხმაურის მოდელირებისათვის. მაგალითად რადიოს მოსმენისას ხშირად გვესმის ხმაური, რომელიც სიგნალს ერთვის და ამახინჯებს. შეიძლება ჩვენც გარკვეული ექსპერიმენტის მოდელირებისთვის სიგნალს დავურთოთ ხმაური უფრო რეალური სურათის შესაქმნელად.

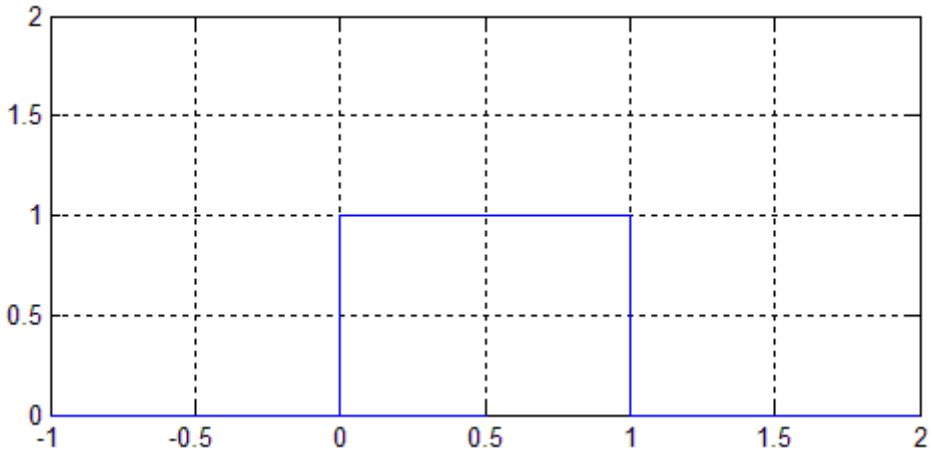
4.2.1 ფუნქციები შემთხვევითი რიცხვებისათვის

ფუნქცია rand MATLAB- ში ქმნის შემთხვევით რიცხვებს ინტერვალში [0, 1], ასე მიღებულ შემთხვევით სიდიდეთა განაწილება თანაბარია. **seed** მნიშვნელობა გამოიყენება ინიციალიზაციისათვის.

rand (n)	ეს ფუნქცია გვაძლევს n სტრიქონიან, n სვეტიან მატრიცას, რომლის ელემენტებიც 0 და 1 შორის მოთავსებული შემთხვევითი რიცხვებია. მიღებული შერჩევის განაწილება თანაბარია
rand (m,n)	გვაძლევს m სტრიქონიან n სვეტიან მატრიცას, რომლის ელემენტებიც 0 და 1 შორის მოთავსებული შემთხვევითი რიცხვებია. მიღებული შერჩევის განაწილება თანაბარია
randn (n)	ეს ფუნქცია გვაძლევს n სტრიქონიან, n სვეტიან მატრიცას, რომლის ელემენტებიც ქმნიან შერჩევას ნორმალური განაწილებით, რომლის საშუალოა 0, ხოლო დისპერსია და სტანდარტული გადახრა 1
randn (m,n)	ეს ფუნქცია გვაძლევს m სტრიქონიან, n სვეტიან მატრიცას, რომლის ელემენტებიც ქმნიან შერჩევას ნორმალური განაწილებით, რომლის საშუალოა 0, ხოლი დისპერსია და სტანდარტული გადახრა 1
rand('seed',s)	განსაზღვრავს seed მნიშვნელობას
rand('seed',0)	დააბრუნებს seed მნიშვნელობას საწყის მდგომარეობაში

4.2.2 შემთხვევით სიდიდეთა თანაბარი განაწილება

შემთხვევითი სიდიდეები ხასიათდება სიმკვრივის ფუნქციით, რომელიც განაწილების ფუნქციის წარმოებულს წარმოადგენს. ეს ფუნქცია ჰგავს ჰისტოგრამას, გვიჩვენებს შემთხვევით სიდიდეთა ინტერვალს და განსაზღვრავს ცალკეული სიდიდეების ხდომილობის ალბათობას. MATLAB ფუნქცია **rand** გვაძლევს შემთხვევით სიდიდეთა ერთობლიობას ისე, რომ თითოეულის მნიშვნელობა თანაბრადალბათურია ინტერვალში [0,1]. მისი სიმკვრივის ფუნქცია მოცემული 5.5 ნახაზზე. სიმკვრივის ფუნქცია უჩვენებს შემთხვევით სიდიდეთა ზედა და ქვედა ზღვარს (0 და 1) x ღერძზე. ფუნქციის (y ღერძი) მართკუთხა ფორმა მიუთითებს, რომ 0 სა და 1 შორის ყველა სიდიდის ხდომილობის ალბათობა ერთნაირია. სიმკვრივის ფუნქციით შემოსაზღვრული ფართი ერთის ტოლია. ამიტომ თუ მისი სიგანე 1-ის ტოლია, სიმაღლეც ერთის ტოლი უნდა იყოს.



ნახ. 5.5 თანაბარი განაწილების სიმკვრივის ფუნქცია

შემდეგი ბრძანებები ქმნის თანაბრად განაწილებული 10 სიდიდის ერთობლიობას, რომელთა მნიშვნელობები 0 და 1 შორისაა:

rand (‘seed’,0) განსაზღვრავს seed საწყის მდგომარეობას
rand(10,1) აწარმოებს 10 სიდიდეს

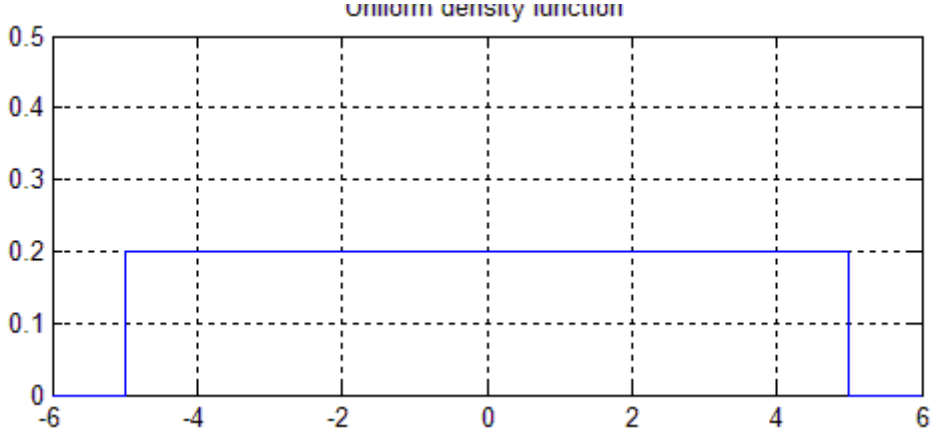
rand ფუნქცია აწარმოებს შემთხვევით სიდიდეთა ერთიდაიგივე მიმდევრობას ერთიდაიგივე ‘seed’ შემთხვევაში. ეს ბრძანება მოგვცემს სვეტ ვექტორს:

0.2190
0.0470
0.6789
0.6793
0.9347
0.3835
0.5194
0.8310
0.0346
0.0535

ხშირად საჭიროა შემთხვევითი სიდიდეები არა მხოლოდ [0 1] ინტერვალში. მაგალითად 5.6 თანაბარი განაწილების ფუნქციაა –5-სა და5 შორის.

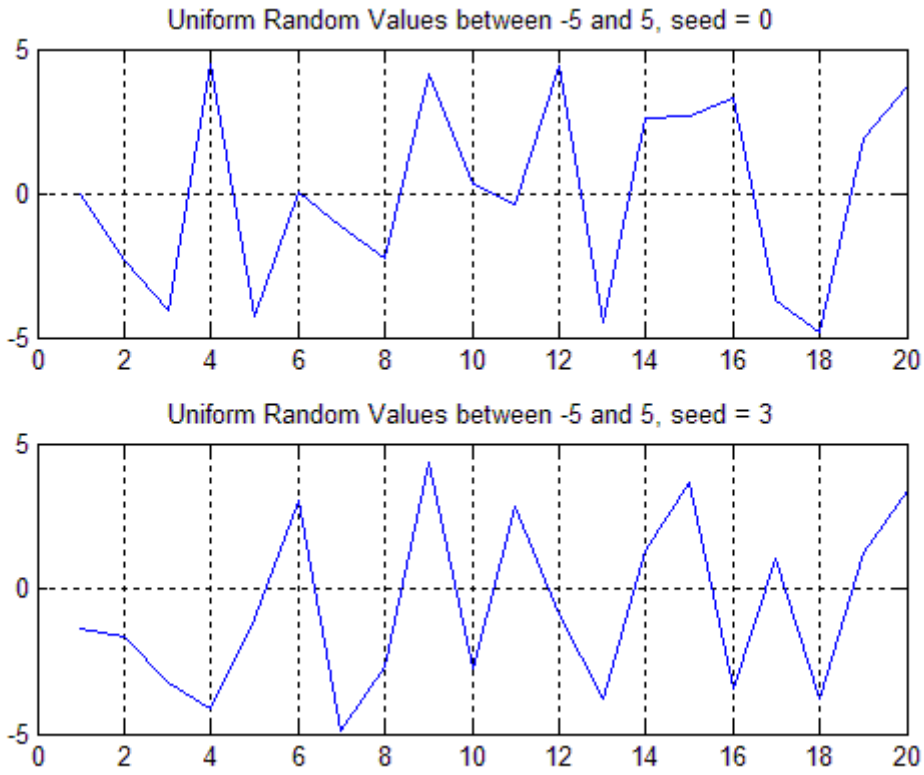
დავუშვათ გვაქვს r შემთხვევითი სიდიდე, რომელიც ვაწარმოეთ თანაბარი განაწილების განერატორით [0 1] შორის. შეგვიძლია ვაწარმოოთ შემთხვევითი სიდიდე, თანაბარი განაწილებით, რომლის მნიშვნელობაც იქნება სხვა საზღვრებში. პირველ რიგში r გავამრავლებთ სიმკვრივის ფუნქციის სიგანეზე. სიგანე გამოითვლება ზედა და ქვედა საზღვრებს შორის სხვაობით. მიღებულ შედგეს დავუმატებთ ქვედა საზღვრის მნიშვნელობას, რომ მივიღოთ საჭირო ინტერვალი. მაგალითად, დავუშვათ გვსურს მივიღოთ შემთხვევითი სიდიდეთა ერთობლიობა ინტერვალში –5, 5. პირველ რიგში ვაწარმოებთ შემთხვევით სიდიდეებს 0-სა და 1 შორის, შემდეგ მათ გავამრავლებთ 10-ზე (5 – (-5)). შემდეგ დავუმატებთ –5 და მივიღებთ შემთხვევით სიდიდეთა ერთობლიობას, რომელიც განაწილებულია თანაბრად ინტერვალში [–5 5]. მაშასადამე, თუ გვსურს [0 1] ინტერვალში თანაბრად განაწილებული სიდიდეები გადავიყვანოთ ისეთ სიდიდეებში, რომელნიც თანაბრად იქნებიან განაწილებულნი ინტერვალში [a b], უნდა ვისარგებლოთ ფორმულით:

$$x = (b - a) * r + a$$



ნახ. 5.6 სიმკვრივის ფუნქცია თანაბარი განაწილებისათვის ინტერვალში –5 და 5

თუ შემთხვევით სიდიდეთა ორი მიმდევრობა ელემენტთა ერთიდაიგივე რაოდენობით და ერთნაირი საზღვრებით ვაწარმოეთ seed სხვადასხვა მნიშვნელობით, შედეგიც სხვადასხვა იქნება. ზოგჯერ საინჟინრო პროგრამაში შესაძლოა დაგვჭირდეს შემთხვევით სიდიდეთა ზუსტად ერთნაირი მიმდევრობის წარმოება სხვადასხვა შემთხვევაში, ამის საშუალებას მოგვცემს seed ერთიდაიგივე მნიშვნელობით სარგებლობა.



ნახ. 5.7 შემთხვევით სიდიდეთა მიმდევრობა ინტერვალში –5 და 5 **seed** სხვადასხვა მნიშვნელობით

სავარჯიშო

მიეცით MATLAB–ს ბრძანება, რომ შექმნათ შემთხვევით რიცხვთა 10 წევრიანი მიმდევრობა შემდეგი მახასიათებლებით:

1. მიმდევრობა თანაბარი განაწილებით ინტერვალში 0 – 10.0.
2. მიმდევრობა თანაბარი განაწილებით ინტერვალში -1 – 1.
3. მიმდევრობა თანაბარი განაწილებით ინტერვალში -20 – 10.
4. მიმდევრობა თანაბარი განაწილებით ინტერვალში 4.5 – 5.
5. მიმდევრობა თანაბარი განაწილებით ინტერვალში π - -π.

შემთხვევით სიდიდეთა მიმდევრობას აგრეთვე უწოდებენ შემთხვევით სიგნალს. ხშირად გვაინტერესებს სიგნალის საშუალო მნიშვნელობა, რომელიც ადვილად გამოითვლება **mean** ფუნქციის საშუალებით. თეორიულად თანაბარი განაწილების საშუალო მნიშვნელობა ტოლია განაწილების ფუნქციის შუაწრტილისა ანუ;

$$\mu = \frac{upper\_bound + lower\_bound}{2}$$

ასევე საინტერესოა სიგნალის დისპერსიის გამოთვლა. ეს დიდიდე შეიძლება გამოვითვალოთ როგორც კვადრატული ფესვი **std** ფუნქციიდან. ალბათობის თეორია უჩვენებს, რომ დისპერსია შემდეგნაირადაც შეგვიძლია გამოვთვალოთ:

$$\sigma^2 = \frac{\sum_{k=1}^{N} x_k^2}{N} - \mu^2 \qquad (5.6)$$

თანაბარი განაწილების გადახრა ასევე ალბათობის თეორიის გამოყენებით სხვაგვარადაც შეგვიძლია გამოვთვალოთ:

$$\sigma^2 = \frac{(upper\_bound - lower\_bound)^2}{12} \qquad (5.7)$$

რადგან ზედა დაქვედა საზღვრებს შორის სხვაობა განაწილების ფუნქციის სიგანეს უდრის:

$$\sigma^2 = \frac{width^2}{12} \qquad (5.8)$$

შევნიშნავთ, რომ 5.6 განტოლება ვრცელდება შემთხვევით სიდიდეთა ყველა სახის მიმდევრობაზე, მაშინ როცა 5.8 – მხოლოდ თანაბარი განაწილებისთვისა ვარგისი. ასევე მნიშვნელოვანია გვახსოვდეს, რომ როცა საქმე გვაქვს რეალურ მონაცემებთან, საშუალო და დისპერსია თეორიულ მნიშვნელობებთან ყოველთვის როდია კარგ თანხვდომაში. თუმცა მოცემულ ინტრვალში მიმდევრობის რაც უფრო მეტი რაოდენობის წევრებს ავიღებთ, გამოთვლილი და თეორიული მნიშვნელობები მით უფრო დაუახლოვდებიან ერთმანეთს. ამის საილუსტრაციოდ შევქმნათ MATLAB საშუალებით შემთხვევით რიცხვთა მიმდევრობა თანაბარი განაწილებით ინტერვალში [-5 5]. თეორიული საშუალო 0 –ის ტოლია, ხოლო თეორიული დისპერსია – 100/12 = 8.333. ცხრილში მოყვანილია მიმდევრობის წევრთა რაოდენობა და MATLAB ფუნქციებით გამოთვლილი საშუალო და დისპერსიის შესაბამისი მნიშვნელობები.

მიმდევრობის წევრთა რაოდენობა	საშუალო	გადახრა
10	0.1286	9.1868
100	0.3421	8.0959
1000	0.0036	7.8525
5000	-0.0022	8.2091

რაც მეტია მოცემულ ინტერვალში შემთხვევით სიდიდეთა რაოდენობა, თეორიული და გამოთვლილი მნიშვნელობები მით მეტად უახლოვდება ერთმანეთს.

თუ **seed** სხვადასხვა მნიშვნელობას ავიღებთ შემთხვევითი სიდიდეთა გენერირებისას, ანდა თუ **rand** ბრძანებას გავიმეორებთ ზედიზედ seed მნიშვნელობის შეუცვლელად, მივიღებთ შემთხვევით სიდიდეთა განსხვავებულ მიმდევრობას.

სავარჯიშო

გამოთვალეთ წინასწარგანსაზღვრული მახასიათებლების მქონე შემთხვევით რიცხვთა მიმდევრობის საშუალო და გადახრა. MATLAB საშუალებით აწარმოეთ განსაზღვრული

მახასიათებლების მქონე 1000 შემთხვევითი რიცხვი. გამოთვალეთ მიღებული განაწილების თეორიული საშუალო და დისპერსია და შეადარეთ იგი შესაბამის თეორიულ მნიშვნელობებს.

1. თანაბარი განაწილება ინტერვალში 0 და 10.
2. თანაბარი განაწილება ინტერვალში –1 და +1.
3. თანაბარი განაწილება ინტერვალში -20 და -10.
4. თანაბარი განაწილება ინტერვალში 4.5 და 5.
5. თანაბარი განაწილება ინტერვალში π და $-\pi$.

4.2.3 ნორმალური ანუ გაუსის განაწილება

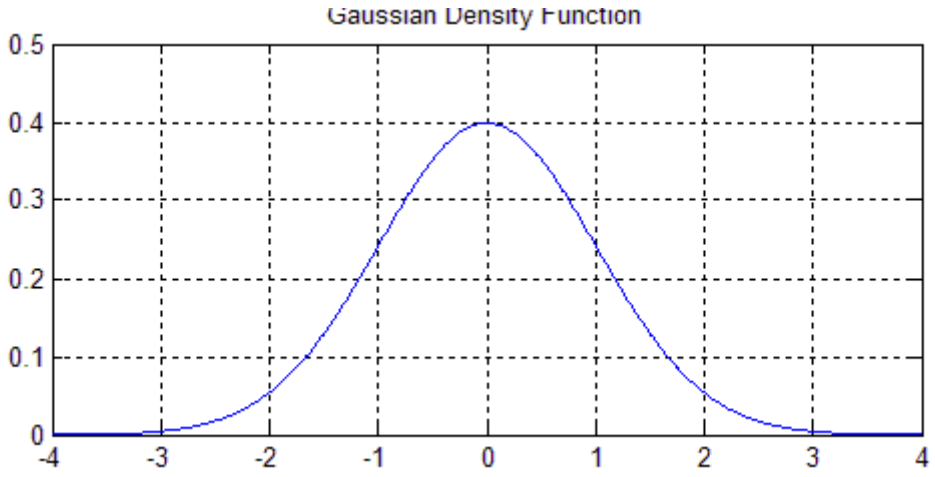
თანაბარი განაწილების დროს ყველა სიდიდის ხდომილობის ალბათობა ერთნაირია. ზოგჯერ გვჭირება შევქმნათ ისეთი განაწილება, სადაც სიდიდეთა რომელიღაც მნიშვნელობები უფრო ხშირად გვხვდება სხვა მნიშვნელობებთან შედარებით. მაგალითად, დავუშვათ შემთხვევით სიდიეთა მიმდევრობა წარმოადგენს გარემოს ტემპერატურის ანათვლებს დროის რომელიმე ინტერვალში. ვნახავთ, რომ ტემპერატურის მნიშვნელობები განსხვავდებიან, მაგრამ არ არიან ერთნაირად ალბათურნი. მაგალითად ტემპერატურა იცვლება მხოლოდ რამდენიმე გრადუსუს ფარგლებში, თუმცა შესაძლოა დიდი ცვლილებები აღინიშნოს ქარის ამოვარდნის, მოღრუბლულობის ანდა დღე – ღამის მონაცვლეობის გამო.

შემთხვევით სიდიდდეთა ასეთი მიმდევრობის მოდელირება შესაძლებელია გაუსის შემთხვევითი ცვლადით (ნორმალური). მისი სიმკვრივის ფუნქცია;

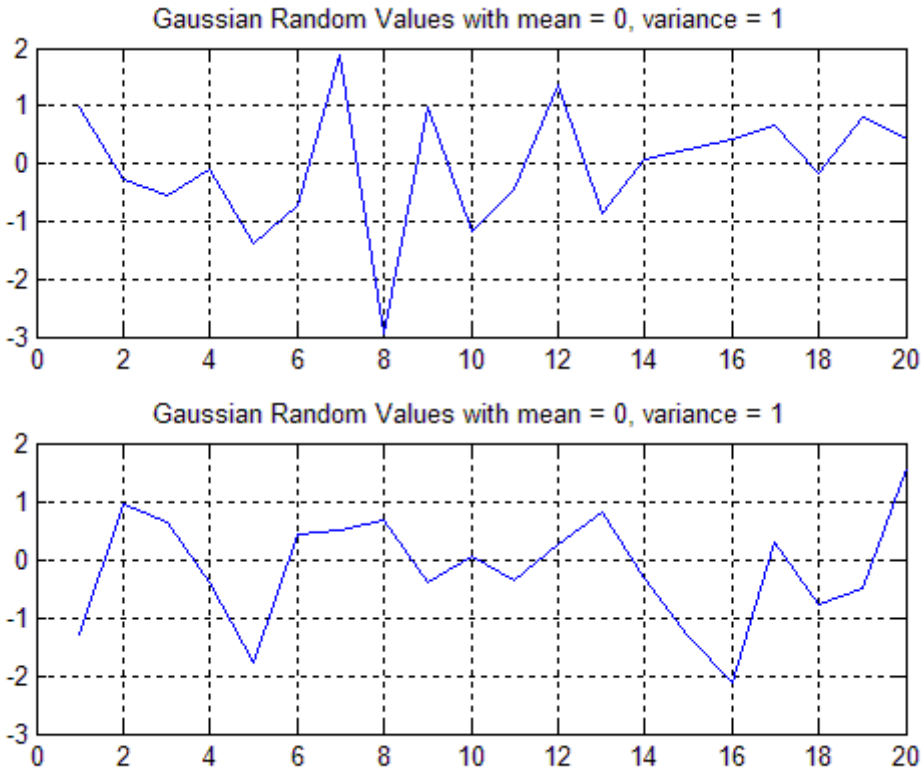
$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma^2)}$$

სადაც μ საშუალოა, ხოლო σ^2 გადახრა. მაგალითად სიმკვრივის ფუნქცია გაუსის განაწილებისათვის , სადაც $\mu=0$ და $\sigma^2 = 1$ ნაჩვენებია ნახ. 5.8.

ასეთი განაწილების საშუალო მნიშვნელობა შეესაბამება სიმკვრივის ფუნქციის მაქსიმუმის წერტილის x კოორდინატს. სიმკვრივის ფუნქციის გრაფიკის მიხედვით შეამჩნევთ, რომ საშუალო მნიშვნელობების მახლობელი სიდიდეების გენერირების ალბათობა უფრო დიდია. შევნიშნავთ, რომ თანაბარ განაწილებას ახასიათებს მნიშვნელობათა ზედა და ქვედა საზღვარი, ხოლო გაუსის განაწილებას ასეთი საზღვრები არ გააჩნია. გაუსის შემთხვევოთი სიდიდეების უმრავლესობა საშუალოდან მცირედ განსხვავებულ მნიშვნელობებს იღებს, თუმცა ზოგიერთი მათგანი შესაძლოა საკმარისად დაშორდეს საშუალო მნიშვნელობას. 5.9 ნახაზი წარმოადგენს ორი განსხვავებული ნორმალური განაწილებას საშუალოთი 0 და გადახრით 1. შესაბამისი განაწილების სიმკვრივის ფუნქციის გარაფიკი მოცემულია ნახ. 5.8.

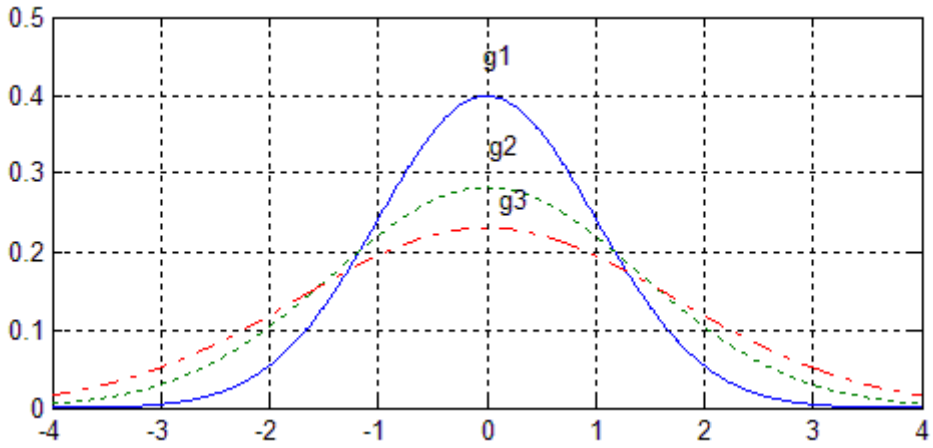


ნახ. 5.8 ნორმალური განაწილების სიმკვრივის ფუნქცია $\mu = 0$, $\sigma^2 = 1$.



ნახ. 5.9 შემთხვევით სიდიდეთა მიმდევრობა გაუსის განაწილებით. $\mu = 0$, $\sigma^2 = 1$.

ნახ 5.10 წარმოდგენილია გაუსის განაწილების სიმკვრივის სამი სხვადასხვა ფუნქცია. სამივე მათგანის საშუალო მნიშვნელობაა 5, მაგრამ მათი სტანდარტული გადახრა და დისპერსია სხვადასხვაა. g1 განაწილებას აქვს ყველაზე მცირე დისპერსია, g3 – ყველაზე დიდი.



ნახ. 5.10 გაუსის სამი სხვადასხვა განაწილების სიმკვრივის ფუნქცია

სტატისტიკიდან ცნობილია, რომ გაუსის განაწილებისათვის სიდიდეთა 68% ღებულობს მნიშვნელობებს საშუალოდან ±σ ინტერვალში, 95% ±2σ ინტერვალში, ხოლო 99% - ±3σ ინტერვალში.

MATLAB ფუნქცია **randn** ქმნის შემთხვევით სიდიდეთა მიმდევრობას საშუალოთი 0 და დისპერსიით 1 (გაუსის, ანუ ნორმალური განაწილებით). იმისათვის რომ შევქმნათ განაწილება განსხვავებული პარამეტრებით, ეს სიდიდე უნდა გავამრავლოთ შესაბამის სტანდარტულ გადახრაზე და დავუმატოთ საშუალო მნიშვნელობა. ამგვარად, თუ r შემთხვევითი სიდიდეა საშუალოთი 0 და სტანდარტული გადახრით 1, შემდეგი განტოლება აწარმოებს შემთხვევით სიდიდეს x საშუალოთი b და სტანდარტული გადახრით a:

$$x = a \cdot r + b$$

შემდეგი ბრძანებები გვაძლევს შემთხვევით სიდიდეთა მიმდევრობას გაუსის ანუ ნორმალური განაწილებით, რომლის საშუალოა 5 და დისპერსია 2:

```
randn ('seed', 0)
s = sqrt(2)*randn(10,1) + 5
```

მივიღებთ ვექტორს s:

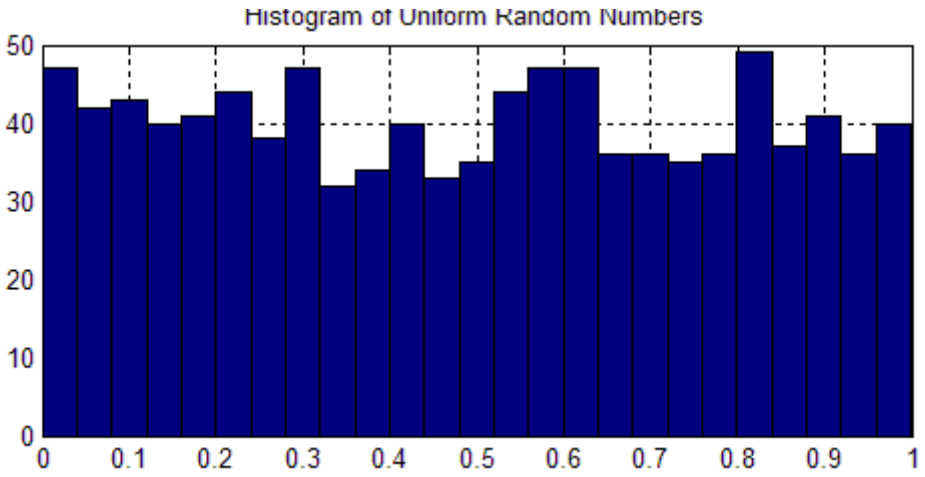
```
s =

    6.6475
    5.8865
    5.1062
    5.4972
    4.0150
    7.3987
    5.0835
    7.5414
    5.3734
    6.2327
```

4.2.4 სიმკვრივის ფუნქცია

განყოფილებაში 5.1 განვიხილეთ ბრძანება **hist**, რომელიც გვაძლევს ჰისტოგრამას. ეს ბრძანება შეიძლება გამოვიყენოთ სიგნალის სიმკვრივის ფუნქციის შესაფასებლად. თუ გვაქვს 1000 შემთხვევითი სიდიდე და ავაგებთ მათ ჰისტოგრამას, ფაქტიურად ჩვენ ვაგებთ მათი განაწილების სიმკვრივის ფუნქციის გრაფიკს. მაგალითად MATLAB საშუალებით შევქმენით თანაბრად განაწილებული 1000 რიცხვის მიმდევრობა 0 და 1 შორის, ჩავწერეთ ისინი სვეტ ვექტორში u_values. შეგვიძლია ვისარგებლოთ **hist** ბრძანებით იმისათვის, რომ ავაგოთ სათანადო განაწილების ფუნქცია 25 სვეტით:

```
u_values = rand(1000,1);
hist(u_values,25)
```

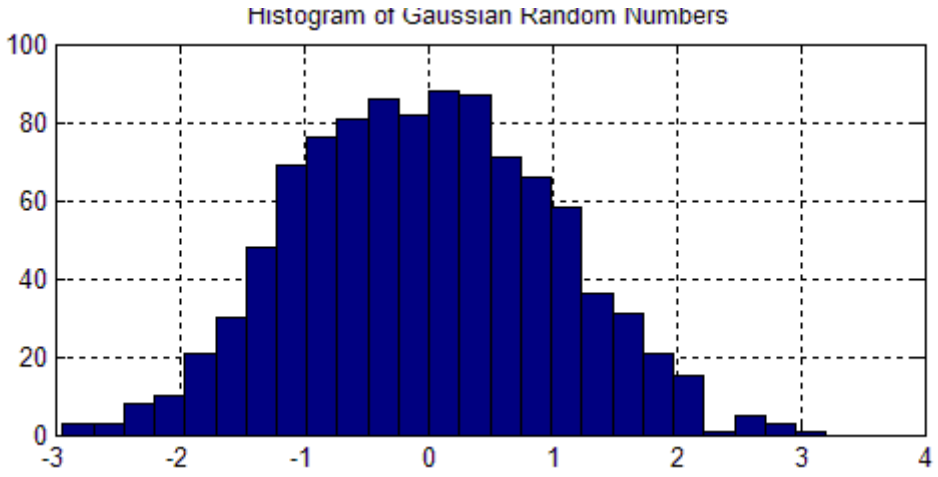
მივიღებთ გრაფიკს 5.11. როგორც მოსალოდნელი იყო, სიდიდეები განაწილებული არიან 0 და 1 შორის და განაწილება შედარებით თნაბარია. ახლა გავიმეოროთ იგივე ნორმალური განაწილებისათვის საშუალოთი 0 და გადახრით 1.



ნახ. 5.11 თანაბარი განაწილების ჰისტოგრამა

```
g_values = randn(1000,1);
hist(u_values,25)
```

მივიღებთ გრაფიკს 5.12. როგორც მოსალოდნელი იყო, განაწილების გრაფიკის პიკია 0, საშუალო მნიშვნელობა და მონაცემთა უმრავლესობის მნიშვნელობები მოთავსებულია ინტერვალში $\pm 2\sigma$.



ნახ. 5.12 ნორმალური განაწილების ჰისტოგრამა

სავარჯიშო

MATLAB საშუალებით აწარმოეთ 1000 შემთხვევითი სიდიდე განსაზღვრული მახასიათებლებით. გამოთვალეთ მისი საშუალო და სტანდარტული გადახრა. გამოთვლილი და მოცემული მნიშვნელობები ერთმანეთთან ახლოს უნდა იყოს. ააგეთ მათთვის ჰისტოგრამა 25 სვეტით:

1. შემთხვევით სიდიდეთა გაუსის განაწილება საშუალოთი 1 და სტანდარტული გადახრით 0.5
2. შემთხვევით სიდიდეთა გაუსის განაწილება საშუალოთი –5.5 და სტანდარტული გადახრით 0.25
3. შემთხვევით სიდიდეთა გაუსის განაწილება საშუალოთი –5.5 და სტანდარტული გადახრით 1.25
4. შემთხვევით სიდიდეთა გაუსის განაწილება საშუალოთი π და სტანდარტული გადახრით $\pi/8$.

პრობლემა: თვითმფრინავის ფრენის მოდელირება

კომპიუტერული მოდელირება გულისხმობს პროგრამულად შეიქმნას სიტუაცია, რომელიც რეალური პროცესების იმიტაციას წარმოადგენს. მოდელირების ნიმუშია მაგალითად კომპიუტერული თამაშები. კომპიუტერული თამაშის დროს თქვენი მოქმედების მიხედვით პროგრამა ირჩევს შესაბამის პასუხს. ზოგიერთი ანიმაციური თამაში იყენებს კომპიუტერულ გრაფიკას იმისათვის, რომ უპასუხოს მოთამაშის რეაქცის, კლავიატურით თუ მაუზით მანიპულირებას. მოდელირების უფრო სრულყოფილ პროგრამებში, როგორიცაა ფრენის სიმულატორი, კომპიუტერი არა მარტო პასუხობს სათანადოდ მომხმაებლის მიერ მიწოდებულ ბრძანებებს, არამედ სიტუაციის შესაბამისად აწარმოებს ისეთ სიდიდეებს, როგორიცაა ტემპერატურა, ქარის სიჩქარე და თვითმფრინავის მდებარეობა. სიმულატორი, ამასთანავე, ახდენს მოულოდნელი მოვლენების მოდელირებას, რომელთაც შესაძლოა ადგილი ჰქონდეს თვითმფრინავის ფრენისას. იმისათვის, რომ პროგრამის მიერ ასახული სიტუაცია, რაც შეძლება ახლოს იყოს რეალურთან, გენერირებული მონაცემები შემთხვევით ხასიათს უნდა ატარებდეს. მონტე კარლოს მეთოდი იყენებს შემთხვევით რიცხვებს მოვლენების მოდელირებისათვის.

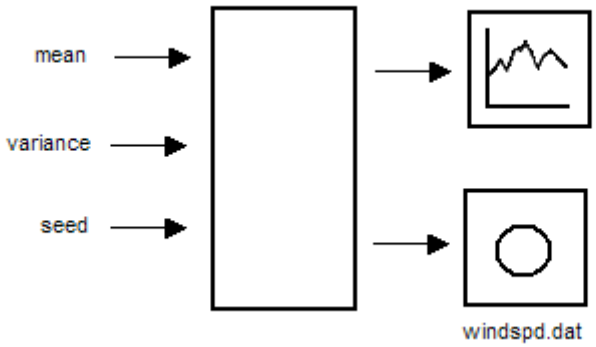
დაწერეთ პროგრამა რომელიც აწარმოებს შემთხვევვით სიდიდეთა მიმდევრობას, იმისათვის, რომ მოხდეს ქარის სიჩქარის მოდელირება, რომლის ხანგრძლივობა იქნება 1 სათი, მონაცემები განახლდება ყოველ 10 წუთში (მივიღებთ 361 მოვაცემს). რელური ექსპერიმენტის მონაცემთა ანალიზის საფუძველზე დადგენილია, რომ ქარის სიჩქარის მოდელირებისათვის შეგვიძლია გამოვიყენოთ ნორმალური შემთხვევითი სიდიდე (გაუსის განაწილება. ჩავთვალოთ, რომ საშუალო და გადახრა მოცემული რეგიონისათვის წელიწადის განსაზღვრულ დროს გარკვეული, დადგენილი, ცნობილი სიდიდეებია და პროგრამაში ვიყენებთ როგოც მიწოდებულ, INPUT პარამეტრებს. ამასთან, შეფასებულია, რომ არსებობს 1% შანსი, ალბათობა იმისა, რომ თვითმფრინავი მოხვდეს მცირედ ქარიშხალში, ისე, რომ ქარის სიჩქარე გაიზარდოს 10 მილი/წმ –ით, ხოლო 0.01% ალბათობა იმისა, რომ მოხვდეს ძლიერ გრიგალში, რომელიც სიჩქარეს 50 მილი/წმ-ით გაზრდის. ააგეთ ქარის სიჩქარის გრაფიკი დროის მიმართ და შეინახეთ მონაცემები ASCII ფორმატით ფაილში სახელით windspd.dat.

1. ამოცანის დასმა

შევქმნათ ერთსაათიანი ქარის მოდელი სტატისატიკური პარამეტრების გათვალისწინებით.

2. INPUT/OUTPUT აღწერა

როგორც ჩანს 5.13 ნახაზიდან, პროგრამის საწყისი (INPUT) მნიშვნელობაა ამინდის სტატისტიკური მონაცემები თვითმფრინავის ფრენის გზაზე: ქარის სიჩქარის საშუალო მნიშვნელობა და სტანდარტული გადახრა ჩვეულებრივ პირობებში. შედეგად კი ვღებულობთ მონაცემთა ფაილს და გრაფიკს, რომელიც ასახავს ქარის სიჩქარის ცვლილებას.



nax. 5.13 I/O დიაგრამა

3. სახელდახელო ამოხსნა

პროგრამა იყენებს შემთხვევით სიდიდეთა სხვადასხვა მიმდევრობას გარკვეული საშუალოთი და დისპერსიით. მცირე ქარიშხლისა და გრიგალის ხდომილების ალბათობა მოცემულია პროცენტებში და წარმოადგებს შემთხვევით სიდიდეებს (განაწილება თანაბარია). გრიგალის მოდელირებისათვის ვაწარმოებთ შეთხვევით სიდიდეს 0 და 1 შორის ბრძანებით **rand** და ვუშვებთ, რომ მცირე ქარიშხალს ადგილი ექნება, თუ მიღებული სიდიდე მოთავსდება საზღვრებში [0.0, 0.01], ხოლო გრიგალს იმ შემთხვევაში ექნება ადგილი, თუ მიღებული სიდიდე მოთავსდება საზღვრებში [0.01 0.0101].

4. MATLAB ამოხსნა

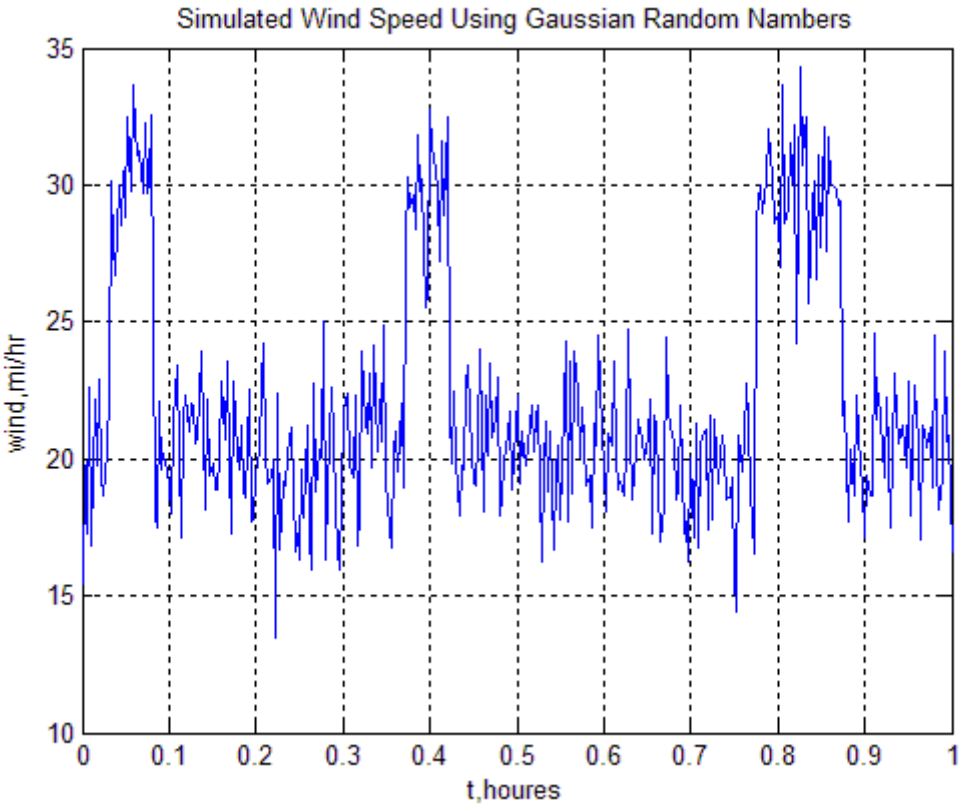
```
%        this program generates one hour of simulated wind
%        such that it uses uniform random nambers to generate
%            the initial wind and compares it to the wind
generating
%        with using Gaussian nambers
%

mn_speed=input('Enter mean of wind speed  ');
var_speed=input('Enter variance of wind speed  ');
std_speed=sqrt(var_speed);
seed=input('Enter seed for random numbers  ');
%
%       generate simulated wind speed without storms
%
rand('seed',seed)
speed=std_speed*randn(1,361)+mn_speed;

%Add simulated storms and microbursts

t=[0:1:360]*(1/360);
k=1;
while k<=361
    random_x=rand(1);
    if random_x<=0.01
        end_storm=min(361,k+17);
        speed(k:end_storm)=speed(k:end_storm)+10;
        k=k+18;
    elseif 0.01<random_x&random_x<=0.0101
        end_micro=min(361,k+5);
        speed(k:end_micro)=speed20(k:end_micro)+50;
        k=k+6;
    else
        k=k+1;
    end
end
%
%                Plot the data
%

    plot(t,speed),...
    title('Simulated  Wind  Speed  Using  Gaussian  Random
Nambers'),...
    xlabel('t,houres'),...
    ylabel('wind,mi/hr'),...
    grid

data(:,1) = t';
data(:,2) = speed';
save windspd.dat data  /ascii
```

5. შემოწმება

თუ პროგრამას გაშვების შემდეგ მივაწოდებთ მნიშვნელობებს:

```
Enter mean of wind speed  20
Enter variance of wind speed  5
Enter seed for random numbers  0
```

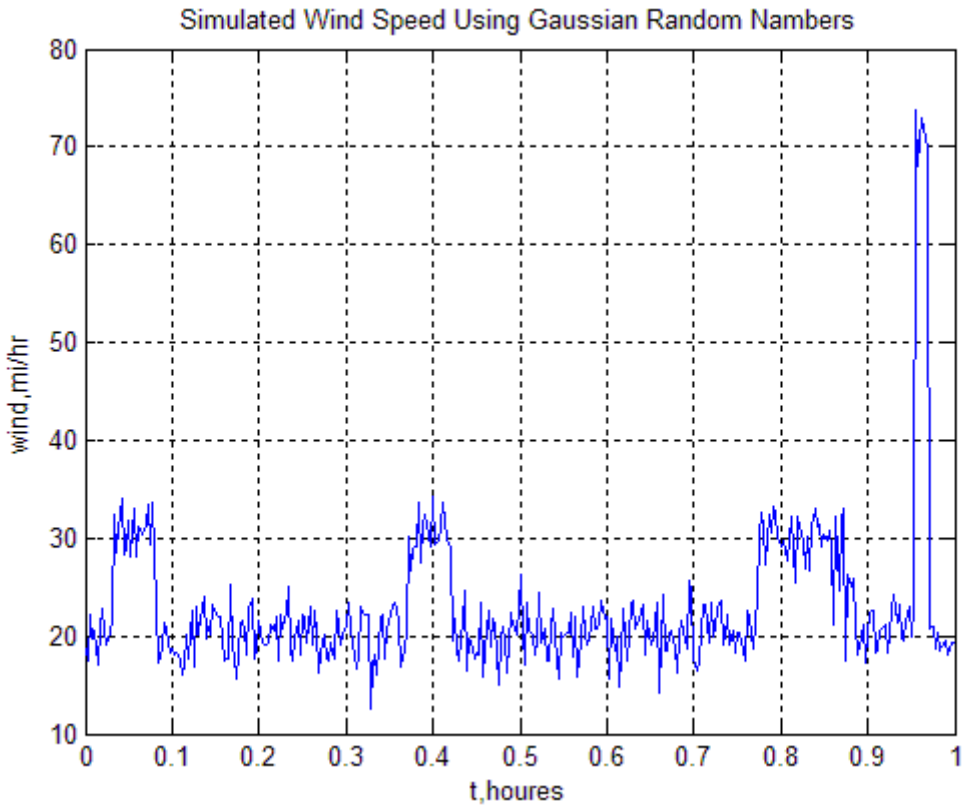
მივიღებთ სურათს ნახ. 5.14



ნახ. 5.15 მოდელირება სამი ქარიშხლით

ქარის უფრო რეალური სურათი რომ მივიღოთ, შემდეგი ცვლილებები შეგვიძლია შევიტანოთ პროგრამაში: გრიგალის განმავლობაში სიჩქარის ნაზრდი შეიძლება იყოს შემთხვევითი სიდიდე თანაბარი განაწილებით საშუალოთი 10 და დისპერსიის დიდი მნიშვნელობით. შეგვიძლია აგრეთვე დავუმატოთ ბრძანება, რომელიც განსაზღვრავს სიჩქარის მომატებას და კლებას არა მკვეთრად, არამედ თანდათანიბით. შეგვიძლია გრიგალის ხანგრძლივობა ავიღოთ თანაბარი განაწილებით შემთხვევითი სიდიდე 1 წუთიდან 10 წუთამდე.

თუ პროგრამას გავუშვებთ რამდენიმეჯერ, **seed** სხვადასხვა მნიშვნელობით, გრაფიკზე გრიგალის შემთხვევასაც მივიღებთ. (ნახ. 5.15).



ნახ. 5.15 ქარის სიჩქარის მოდელირება (გრიგალის შემთხვევა)

1.3 თანაფარდობა სიგნალი/ხმაური

საცდელი საინჟინრო ამოცანების მოდელირებისას ხშირად საჭიროა ისეთი სიგნალის მოდელირება, რომელსაც ახლავს შემთხვევითი ხმაური. (ნახ. 5.16). ხმაური შემთხვევით სიდიდეთა ერთობლიობაა და მისი წილი მიღებულ საგნალში განისაზღვრება თანაფარდობით სიგნალი/ხმაური – SNR. იგი სიგნალის სიმძლავრის ერთეულებში გამოისახება. ჯერ განვიხილოთ რას წარმოადგენს სიგნალის სიმძლავრე და შემდეგ დავუბრუნდეთ SNR.

4.3.1 სიგნალის სიმძლავრე

სიმძლავრე – ეს არის სიგნალის ამპლიტუდის ზომა. რაც მეტია სიგნალის ამოლიტუდა, მით მეტია მისი სიმძლავრე. რამდენადაც ამპლიტუდა შეიძლება იყოს დადებითიც და უარყოფითიც, სიმძლავრე განისაზღვრება ამპლიტუდის კვადრატით, ისე რომ სიმძლავრის მნიშვნელობა ყოველთვის დადებითი სიდიდეა. x ვექტორით წარმოდგენილი სიგნალის სიმძლავრე შეიძლება შევაფასოთ სიგნალის მნიშვნელობათა კვადრატების საშუალო მნიშვნელობით:

$$power \approx \frac{\sum_{1}^{N} x_k^2}{N}$$

MATLAB –ში ეს სიდიდიე გამოითვლება **sum** ფუნქციის სასუალებით:

power = sum(x^2)/N;

შესაძლებელია აგრეთვე ვაჩვენოთ, რომ სიგნალის სიმძლავრე ტოლია დისპერსიისა და საშუალო მნიშვნელობის კვადრატის ჯამისა:

$$power = \sigma^2 + \mu^2$$

MATLAB საშუალებით ეს შეიძლება შემდეგი ბრძანებებით გამოითვალოს:

power = std(x)^2 + mean(x)^2;

თუ სიგნალი სინუსოიდაა, ადვილად შეიძლება ვაჩვენოთ, რომ მისი სიმძლავრე ტოლია მისივე ამპლიტუდის კვადრატის ნახევრის. მაგალითად სინუსოიდის 4sin2πt სიმძლავრე ტოლია 16/2, ანუ 8.

სავარჯიშო

მოცემული პარამეტრების საშუალებით აწარმოეთ 100 შემთხვევითი სიდიდე. გამოთვალეთ სიმძლავრე ორი სხვადასხვა გზით (მნიშვნელობათა კვადრატების შუალო და საშუალოსა და დისპერსიის მნიშვნელობათა მიხედვით) და შეადარეთ ერთმანეთს.

1. თანაბარად განაწილებული შემთხვევითი სიდიდეები ინტერვალში 0 და 10.
2. თანაბარად განაწილებული შემთხვევითი სიდიდეები ინტერვალში -2 და 4.
3. თანაბარად განაწილებული შემთხვევითი სიდიდეები საშუალოთი 0 და გადახრით 1.0
4. თანაბარად განაწილებული შემთხვევითი სიდიდეები საშუალოთი –0.5 და დისპერსიით 4.0
5. შემთხვევოთი სიდიდეები ნორმალური განაწილებით საშუალოთი 0 და დისპერსიით 2.0
6. შემთხვევოთი სიდიდეები ნორმალური განაწილებით საშუალოთი 0 და დისპერსიით 0.5
7. შემთხვევოთი სიდიდეები ნორმალური განაწილებით საშუალოთი –2.5 და დისპერსიით 0.5

4.3.2 SNR გამოთვლა

თანაფარდობა სიგნალი – ხმაური არის სიგნალის სიმძლავრის თანაფარდობა ხმაურის სიმძლავრესთან. მაგალითად, SNR = 1, ნიშნავს, რომ სიგნალს სიმძლავრესა და ხმაურის სიმძლავრის თანაფარდობა = 1:1. თუ მოცემული გვაქვს სინუსოიდა ამპლიტუდით 3 და თანაბრი ხმაურით –1 და 1 შორის, შეგვიძლია გამოვთვალოთ SNR შესაბამის სიმძლავრეთა გამოთვლებზე დაყრდნობით:

$$SNR = \frac{9/2}{1/3} = 13.5$$

თუ მოცემული გვაქვს ძირითადი სინუსოიდური სიგნალი ამპლიტუდით A და თანაბრი ხმაური ინტერვალში [a b], MATLAB საშუალებით SNR ასე გამოითვლება

SNR = ((A^2)/2)/((b-a)^2/12

საილუსტრაციოდ დავუშვათ გვსურს შევქმნათ 201 ელემენტიანი ვექტორი სიგნალი, რომელიც შეიცავს 1 ჰერციან სინუსოიდას ხმაურით, რომლის საშუალოა 0 ისე, რომ SNR = 46. სინუსოიდას უნდა ჰქონდეს ამპლიტუდა 1.5 და ფაზური კუთხე 0.0, ათვლილი უნდა იყოს 100 ჰერცით (ეს ნიშნავს, ათვლა წარმოებს ყოველ 1/100 წამში):

$$\text{SNR} = \text{სიგნალის სიმძლავრე/ხმაურის სიმძლავრე} = ((1.5^2)/2)/\text{ხმაურის სიმძლავრე} = 46$$

აქედან მივიღებთ:

$$\text{ხმაურის სიმძლავრე} = (1.5^2)/2/46 = 0.024$$

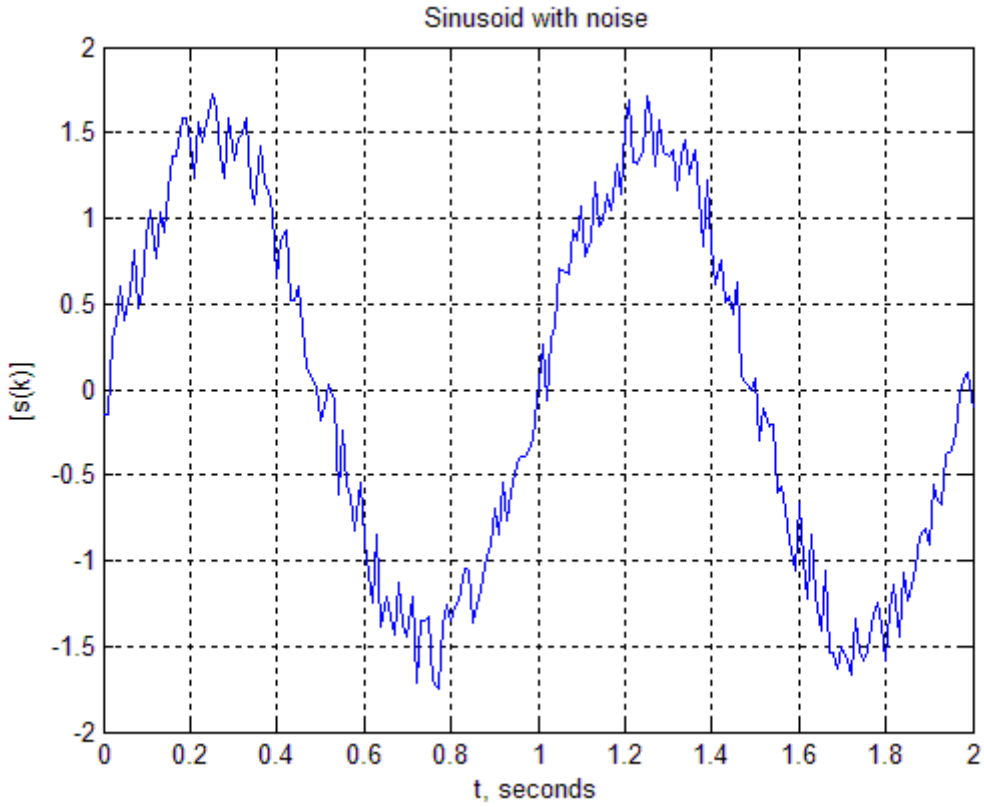
რადგანაც ხმაურის საშუალო = 0, ხმაურის სიმძლავრე ტოლია მისი დისპერსიისა. ე.ი ხმაურის გადახრა ტოლი ყოფილა 0.024, რადგან ხმაური თანაბარია და მისი საშუალო 0 – ის ტოლია, მისი მნიშვნელობები უნდა მდებარეობდეს [a –a] შორის. როგორც ვიცით, დისპერსია ტოლია $(2a)^2/12$, გვექნება:

$$0.024 = (2a)^2/12$$

ანუ a=0.27.

ახლა შეგვიძლია გამოთვლილი აღნიშნული პარამეტრებით შევქმნათ ხმაურის მნიშვნელობები და დავუმატოთ სინუსოიდურ სიგნალს:

```
%
%       Generate and plot sine plus noise.
%
rand('seed',0);
t=(0:0.01:2);
s=1.5*sin(2*pi*t) + (0.54*rand(1,201) -0.27);
plot(t,s),...
    title('Sinusoid with noise'),...
    xlabel('t, seconds'),...
    ylabel('[s(k)]'),...
    grid
```



ნახ. 5.16 სინუსოიდური სიგნალი ხმაურით.

სინუსოიდური სიგნალი თანაბარი ხმაურით ნაჩვენებია გრაფიკზე 5.16. შევნიშნავთ, რომ გრაფიკი მოიცავს სინუსოიდის ორ პეროოდს 2 წამის განმავლობაში, რაც შეესაბამება 1 ჰერციან სინუსოიდას, ესე იგი პერიოდი = 1 წამს.

4.3.3 არსებულ სიგნლზე ხმაურის დამატება

დავუშვათ გვაქვს სიგნალი, რომელიც ათვლილია და შენახულია მონაცემთა ფაილის სახით. თუ გვსურს დავუმატოთ ასეთ სიგნალს ხმაური, რომელიც მოგვცემს SNR განსაზღვრულ მნიშვნელობას, ჯერ უნდა შევაფასოთ სიგნალის სიმძლავრე, იმისათვის, რომ განვსაზღვროთ ხმაურის სიგნალის შესაბამისი სიმძლავრე. სიგნალის სიმძლავრის შესაფასებლად გამოვთვალოთ სიგნალის მნიშვნელობათა კვადრატების საშუალო, რომელიც MATLAB საშუალებით ადვილად გამოითვლება. ამის შემდეგ შეგვიძლია განვსაზღვროთ ხმაურის სიგნალის სიმძლავრე. ვიცით, რომ სიმძლავრე საშუალოსა და დისპერსიის ფუნქცია ამიტომ ერთ-ერთი მათგანი მაინც უნდა ვიცოდეთ, რომ მეორე განვსაზღვროთ. სასურველია, რომ ხმაურის საშუალო მნიშვნელობა 0 –ის ტოლი იყოს, ამ დაშვებას ხშირად მიმართავენ, თუ სხვა ინფორმაცია არ არსებობს. უკვე შეგვიძლია გამოვთვალოთ დისპერსია, მივიღოთ ვექტორი, რომელიც ხმაურის მნიშვნელობებს შეიცავს და დავუმატოთ იგი ძირითად სიგნალს.

ააგეთ სიგნალი, რომელიც შედგება 5 ჰერციანი სინუსოიდის 100 წერტილისაგან ხმაურით, რომლის საშუალოა 0 და ხასიათდება პარამეტრებით:

5.1 თანაბარი ხმაური, SNR =5
5.1 თანაბარი ხმაური, SNR =1
5.1 თანაბარი ხმაური, SNR =0.2
5.1 ხმაური გაუსის განაწილებით, SNR =5
5.1 ხმაური გაუსის განაწილებით, SNR =1
5.1 ხმაური გაუსის განაწილებით, SNR =0.2

ამ თავში წარმოგიდგინეთ მონაცემთა სტატისტიკური ანალიზისათვის საჭირო მთელი რიგი ფუნქციებისა. განვიხილეთ აგრეთვე მაგალითი თუ როგორ შევქმნათ ზოგიერთი საინჟინრო პრობლემის მოდელირებისათვის საჭირო მონაცემები და მივცეთ მათ სტატისტიკური შეფასება. მაგალითებში განხილული იყო საკომუნიკაციო სიგნალის ანალიზი, მოდელირება მონტე-კარლოს მეთოდით და სინალი – ხმაური თანაფარდობის შეფასება.

ბრძანებები და ფუნქციები

cumprod	განსაზღვრავს კუმულაციურ ნამრავლს
cumsum	განსაზღვრავს კუმულაციურ ჯამს
hist	აგებს ჰისტოგრამას
max	განსაზღვრავს უდიდეს მნიშვნელობას
mean	განსაზღვრავს საუალო მნიშვნელობას
median	განსაზღვრავს მედიანას მნიშვნელობას
min	განსაზღვრავს უმცირეს მნიშვნელობას
prod	განსაზღვრავს სიდიდეთა ნამრავლს
rand	განსაზღვრავს შემთხვევით რიცხვებს
sort	დაალაგებს სიდიდეებს
std	გამოითვლის სტანდარტულ გადახრას
sum	განსაზღვრავს სიდიდეთა ჯამს

ამოცანები

ამოცანები 1-7 დაკავშირებულია საინჟინრო პრობლემებთან, რომლებიც ამ თავშია განხილული, ხოლო დანარჩენი ამოცანები სხვა საკითხებს ახება.

საკომუნიკაციო სიგნალი. ეს საკითხები დაკავშირებულია ამ თავში განხილულ ამოცანასთან საკომუნიკაციო სიგნალის შესახებ.

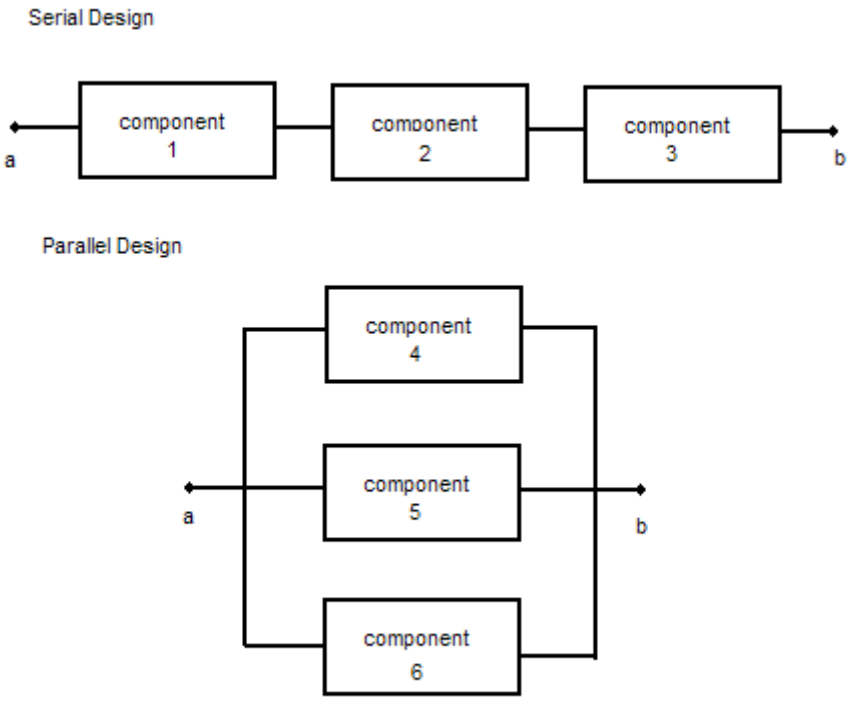
1. მიკროფონის საშუალებით ჩაწერეთ თქვენს მიერ წარმოთქმული სიტყვები – ნული, ერთი, ორი, სამი, . . . ცხრა ცალკე ფაილების სახით. გაუშვით პროგრამა, რომელიც დაწერეთ ამ თავში განხილული ამოცანისათვის საკომუნიკაციო საგნალის შესახებ თითოეული ფაილის მონაცემებისათვის და დაბეჭდეთ შესაბამისი ცხრილები.

2. გაუშვით იგივე პროგრამა სხვადასხვა ადამიანის მიერ წარმოქმნული ერთიდაიგივე სიტყვის შესაბამისი ფაილებისათვის და დაბეჭდეთ ცხრილი შესაბამის სტატიტიკურ მონაცემთა შესადარებლად.

თვითმფრინავის ფრენის მოდელირება. ქარის სიჩქარე. ეს პრობლემებიც ასევე ამთავში განხილულ პრობლემას უკავშირდება ქარის მოდელირებასთან დაკავშირებით.

3. ამ თავში ქარის მოდელირებასთან დაკავშირებით დაწერილი პროგრამა ისე შეცვალეთ, რომ მან ქარის სიჩქარის საწყისი მნიშვნელობებისათვის გამოიყენოს თანბრად განაწილებული შემთხვევითი სიდიდეები. შეადარეთ შედეგები შემთხვევით სიდიდეთა ორი სხვადასხვა ტიპის განაწილების შემთხვევაში.
4. იგივე პროგრამა შეცვალეთ ისე, რომ გრიგალის დროს სიჩქარის მატება იყოს თანაბარი შემთხვევითი სიდიდე საშუალოთი 10 მილი/საათში და სტანდარტული გადახრით 1.5 მილი/საათში.
5. იგივე პროგრამა შეცვალეთ ისე, რომ გრიგალის დროც სიჩქარის მატება იყოს თანაბარი შემთხვევითი სიდიდე, რომლის საშუალოა 10 მილი/საათში და გადახრა ორჯერ აგემატებოდეს გადახრას, რომელიც ახასიათებს ქარის სიჩქარეს გრიგალის გრეშე.
6. იგივე პროგრამა შეცვალეთ ისე, რომ საჭირო იყოს პროგრამისათვის გრიგალის და ქარიშხლის ხდომილებათა ალბათობის მითითება.
7. იგივე პროგრამა შეცვალეთ ისე, რომ გრიგალის ხანგრძლივობა იყოს შემთხვევითი სიდიდე ინტერვალში 1 – 10 წუთი.

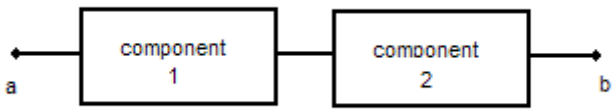
კომპონენტთა საიმედოობა. კომპონენტთა საიმედოობის ანალიზის განტოლებები გამომდინარეობს ალბათობის და სტატისტიკის თეორიიდან. საიმედოობა ასევე შეგვიძლია განვსაზღვროთ კომპიუტერული მოდელირების საფუძველზე, თუ თითოეული ელემენტის საიმედოობა ცნობილი სიდიდეა. განვიხილოთ დიაგრამა ნახ. 5.17. მიმდევრობითი შეერთების ამ სქემაში შეერთებულია 3 კომპონენტი. იმისათვის, რომ ინფორმაცია a წერტილიდან b –ში მივიდეს, სამივე კომპონენტი უნდა მუშაობდეს. პარალელური შეერთების შემთხვევაში სამიდან ერთი მაინც უნდა მუშაობდეს იგივე ინფორმაციის გადასაცემად. თუ ვიცით კომპონენტის საიმედოობა (რომელიც პროპორციულია იმ დროისა, როცა იგი სწორად მუშაობს), შეგვიძლია დავწეროთ განტოლება, რომელიც გამოითვლის მთელი სისტემის საიმედოობას. სისტემის საიმედოობა შეგვიძლია შევაფასოთ კომპიუტერული მოდელირების საფუძველზე. მაგალითად თუ მიმდევრობით ჩართული სისტემის ელემენტთა საიმედოობა 0.8 ტოლია (რაც ნიშნავს, რომ კომპონენტი დროის 80% სწორად მუშაობს), შეგვიძლია ვაწარმოოთ 3 შემთხვევითი სიდიდე 0 - 1 ინტერვალში. თუ სამივე მათგანი ნაკლებია ან ტოლია 0.8, მაშინ სისტემა იმუშავებს. თუ რომელიმე მათგანი მეტია 0.8, მაშინ კომპონენტი არ იმუშავებს (for one simulation). თუ ასეთ სიმულაციას 100 და 1000 ჯერ გავიმეორებთ, შევძლებთ გამოვთვალოთ წილი იმ დროისა, როცა სისტემა მუშაობს. იმისათვის, რომ შევაფასოთ პარალელური სისიტემის



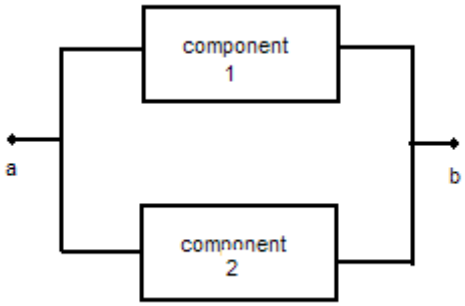
ნახ. 5.17

მუშაობის საიმედოობა, იმგვარადვე ვიქცევით. თუ ამ სამი რიცხვიდან ერთი მაინც ნაკლებია ან ტოლია 0.8, სისტემა იმუშავებს. თუ სამივე შემთხვევითი სიდიდე აჭარბებს 0.8, მაშინ სისტემა არ იმუშავებს.

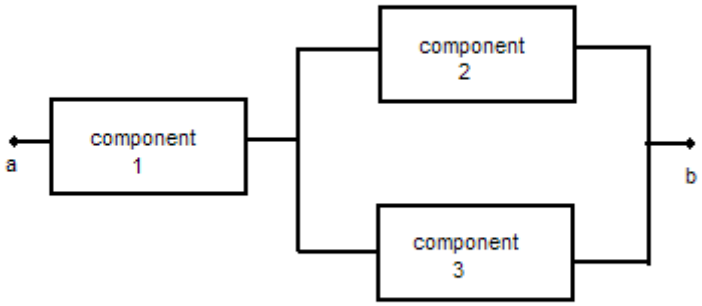
8. დაწერეთ პროგრამა მომდევრობით შეერთებული სისტემის მოდელირებისათვის (ნახ. 5.17), ჩათვალეთ, რომ კომპონენტების საიმედოობაა (reliability) 0.8. გამოითვალეთ საიმედოობა 500 სიმულაციისათვის, 1000 სიმულაციისთვის
9. დაწერეთ პროგრამა პარალელურად შეერთებული სისტემის მოდელირებისათვის (ნახ. 5.17), ჩათვალეთ, რომ კომპონენტების საიმედოობაა (reliability) 0.8. გამოითვალეთ საიმედოობა 500 სიმულაციისათვის, 1000 სიმულაციისთვის. რომელი სისტემა უფრო საიმედოა?
10. დაწერეთ პროგრამა ნახ. 5.18 მოცემული სისტემის მოდელირებისათვის, თუ ერთი კომპონენტის საიმედოობაა 0.8, ხოლო მეორის – 0.92. დაბეჭდეთ საიმედოობა 5000 სიმულაციის შემთხვევაში.
11. დაწერეთ პროგრამა ნახ. 5.19 მოცემული სისტემის მოდელირებისათვის, თუ ერთი კომპონენტის საიმედოობაა 0.8, ხოლო მეორის – 0.92. დაბეჭდეთ საიმედოობა 5000 სიმულაციის შემთხვევაში. შეადარეთ პარალელური მიმდევრობით ჩართული სისტემის საიმედოობა.
12. დაწერეთ პროგრამა ნახ. 5.20 მოცემული სისტემის მოდელირებისათვის, თუ ერთი კომპონენტის საიმედოობაა 0.8, მეორის – 0.85, ხოლო მესამის – 0.95. დაბეჭდეთ საიმედოობა 5000 სიმულაციის შემთხვევაში.



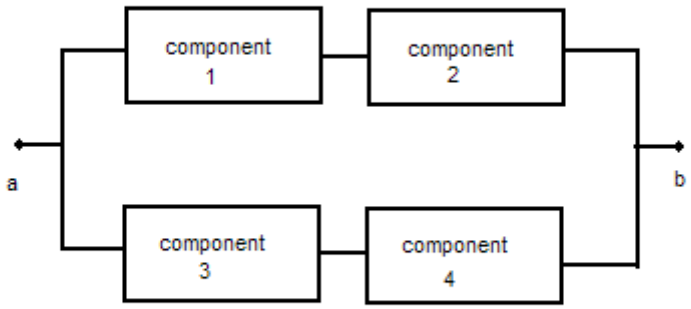
ნახ. 5.18



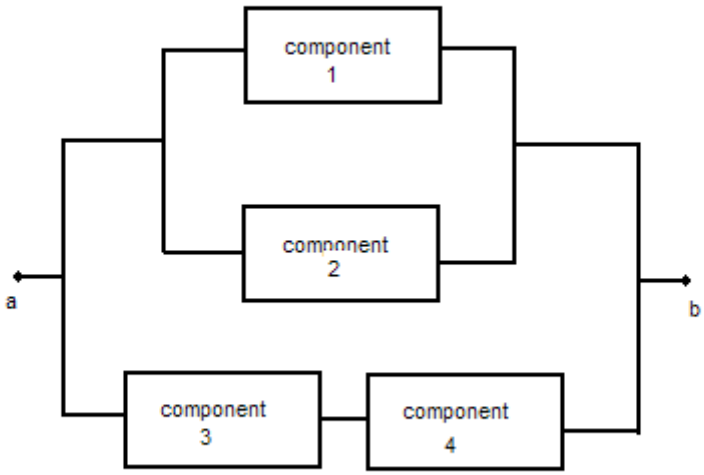
ნახ. 5.19



ნახ. 5.20



ნახ. 5.21



ნახ. 5.22

13. დაწერეთ პროგრამა ნახ. 5.21 მოცემული სისტემის მოდელირებისათვის, თუ 1და 2 კომპონენტის საიმედოობაა 0.8, მესამის – 0.95. დაბეჭდეთ საიმედოობა 5000 სიმულაციის შემთხვევაში.
14. დაწერეთ პროგრამა ნახ. 5.22 მოცემული სისტემის მოდელირებისათვის, კომპონენტების საიმედოობაა 0.95. დაბეჭდეთ საიმედოობა 5000 სიმულაციის შემთხვევაში.

ელექტროსადგურის მიერ გამომუშავებული ენერგია. 8 კვირის განმავლობაში გროვდებოდა მონაცემები ელექტროსადგურის მიერ გამომუშავებული ენერგიეს შესახებ. მონაცემები

ჩაიწერა ფაილში plant.dat. თითოეული სტრიქონი შეიცავს მონაცემებს 1 კვირისათვის. 1, 2, 3 , . . . , 7 დღეს გამომუშავებულ ენერგიას მეგავატებში.

15. დაწერეთ პროგრამა, რომელიც წაიკითხავს მონაცემებს ფაილიდან plant.dat, და მოგვცემს იმ დღეების რაოდენობას, როცა ელექტროსადგურმა გამოიმუშავა საშუალოზე მეტი ენერგია. უნდა მივიღოთ კვირის ნომერი და დღის ნომერი თითოეული ასეთი დღისათვის. ამასთან ერთად დაგვიბეჭდოს ამ რვა კვირის განმავლობაში საშუალოდ ერთ დღეში გამომუშავებული ელექტროენერგია მეგავატებში
16. დაწერეთ პროგრამა, რომელიც წაიკითხავს მონაცემებს ფაილიდან plant. dat, და დაბეჭდავს იმ კვირის და დღის ნომერს, როცა ყველაზე მეტი და ყველაზე მცირე ელექტროენერგია იქნა გამომუშავებული.
17. დაწერეთ პროგრამა, რომელიც წაიკითხავს მონაცემებს ფაილიდან plant. dat, და დაბეჭდავს ერთ დღეში გამომუშავებულ საშუალო ენერგიეს, ყოველი კვირისათვის ცალკ-ცალკე.

4 მატრიცული ოპერაციები

პრობლემა: მანქანური ხედვა

საკომუნიკაციო სისტემას და მიკროპროცესორულ მაკონტროლებელ ინსტრუმენტს შეუძლია მიიღოს მაღალი გარჩევის გამონასახი. ერთ-ერთი მათგანი იუპიტერის "მთვარე" - ჰანიმედი - ნაჩვენებია ფოტოზე, რომელიც გადაიღო კოსმოსურმა თანამგზავრმა 'Galileo'. ფოტოგამონასახის კომპიუტერული ინტერპრეტაცია საკმაოდ რთულია. კარგი ალგორითმის დაწერა მოითხოვს საფუძვლიან ცოდნას გამონასახის შესახებ. გამონასახზე მუშაობისას ზოგჯერ უნდა შეგვეძლოს გავყვეთ მიძრავი ობიექტის კვალს ერთი გამონასახიდან მეორეზე, იმისათვის რომ დავადგინოთ ამ ობიექტის მოძრაობის სიჩქარე და მიმართულება. ასეთი ტიპის ამოცანების ალგორითმი რთული შესადგენია.

შესავალი

საინჟინრო მონაცემების წარმოსადგენად მატრიცა ძალზე მოხერხებული ფორმაა. წინა თავში განვიხილეთ მათემატიკური გამოთვლები და ფუნქციები, რომლებიც გამოიყენება მატრიცების შესაბამის ელემენტებს შორის მათემატიკური ოპერაციების შესასრულებლად. ამ თავში განვიხილავთ მათემატიკურ ოპერაციებს უშუალოდ მატრიცებს შორის. პირველ რიგში განვიხილავთ მათემატიკურ ოპერაციებს რომელთა საშუალებით მატრიციდან ან მატრიცებიდან ახალ მატრიცას ვღებულობთ. შემდეგ გაგაცნობთ ფუნქციებს, რომლებიც მატრიცების მანიპულირების საშუალებას იძლევა და იგი ერთი ფორმიდან მეორეში გადაჰყავს.

4.1 მატრიცული ოპერაციები

საინჟინრო ამოცანებში მატრიცები გამოიყენება როგორც მოხერხებული საშუალება მონაცემების წარმოსადგენად. ამ განყოფილებაში განვიხილავთ გამოთვლებს, რომელშიც მონაწილეობს მატრიცის სახით წარმოდგენილი მონაცემები. ამ თავში ძირითადად შევეხებით მატრიცებს, რომლებიც შეიცავენ ორზე მეტ სვეტს და სტრიქონს. გავიხსენოთ, რომ სკალარული გამრავლება და მეტრიცების შეკრება – გამოკლება წარმოებს შესაბამის ელემენტებს შორის ოპერაციების შესრულებით. ამ თავში მატრიცების გამრავლებას განვიხილავთ, ხოლო გაყოფას თავში, სადაც საუბარი იქნება წრფივ განტოლებათა სისტემის ამოხსნაზე.

4.1.1 მატრიცის ტრანპონირება

ტრანსპონირებული მატრიცა ეს არის ახალი მატრიცა, რომლის სვეტებიც საწყისი მატრიცის სტრიქონებია. ტრანსპონირებულს აღნიშნავენ ნიშნით " ′ "
მაგალითად განვიხილოთ მატრიცა A და მისი ტრანსპონირებული A′:

$$A = \begin{matrix} 2 & 5 & 1 \\ 7 & 3 & 8 \\ 4 & 5 & 21 \\ 16 & 13 & 0 \end{matrix} \qquad A' = \begin{matrix} 2 & 7 & 4 & 16 \\ 5 & 3 & 5 & 13 \\ 1 & 8 & 21 & 0 \end{matrix}$$

თუ დავაკვირდებით ვნახავთ,რომ ელემენტი (3,1) გარდაისახა ელემენტში (1,3) ანუ A(3,1) = A′(1,3), ხოლო A(4,2) = A′(2,4). ფაქტიურად ინდექსები იცვლება ისე, რომ A(i,j) = A′(j,i).

უნდა გვახსოვდეს, რომ თუ მატრიცა არ არის კვადრატული (კვადრატული მატრიცის სვეტების და სტრიქონების რაოდენობა ერთნაირია), მაშინ მატრიცა და მისი ტრანსპონირებული სხვადასხვა ზომისაა. ტრანსპონირების ოპერაციას ხშირად ვიყენებთ, როცა გვსურს სტრიქონი ვექტორიდან მივიღოთ სვეტი ვექტორი.

თუ A მატრიცა შეიცავს კომპლექსურ რიცხვებს, მაშინ A′ მოგვცემს ტრანსპონირებულ მატრიცას, მაგარმ მისი ელემენტები იქნება საწყისი მატრიცის ელემენტების კომპლექსური შეუღლებული, ამიტომ, თუ გვსურს ტრანსპონირებული მატრიცის მიღება ისე, რომ მისი ელემენტების მნიშვნელობა არ შეიცვალოს, უნდა ვისარგებლოთ ბრძანებით A.′ ან conj(A′).

მაგალითად:

```
>> Z=[1+2i, 2+4i, 2+4i; 2i, 6i, 7i];
>> Z'

ans =

   1.0000 - 2.0000i        0 - 2.0000i
   2.0000 - 4.0000i        0 - 6.0000i
   2.0000 - 4.0000i        0 - 7.0000i

>> Z.'

ans =
```

```
1.0000 + 2.0000i        0 + 2.0000i
2.0000 + 4.0000i        0 + 6.0000i
2.0000 + 4.0000i        0 + 7.0000i
```

4.1.2 სკალარული ნამრავლი

ორი ერთნაირი ზომის ვექტორის სკალარული ნამრავლი არის სკალარი რომელიც ტოლია ამ ვექტორების შესაბამის ელემენტთა ურთიერთნამრავლთა ჯამისა. მაგალითად, თუ A და B N ელემენტიანი ვექტორებია, მაშინ მათი სკალარული ნამრავლი ასე გამოითვლება:

$$dot\_product = A \cdot B = \sum_{i=1}^{N} a_i b_i$$

საილუსტრაციოდ დავუშვათ, A და B შემდეგი ვექტორებია:

```
A = [4 –1 3]     B = [-2 5 2]
```

მაშინ სკალარული ნამრავლი იქნება:

$$A \cdot B = 4 \cdot (-2) + (-1) \cdot 5 + 3 \cdot 2 = (-8) + (-5) + 6 = -7$$

სკალარულ ნამრავლს სხვანაირად შიდა ნამრავლს (**inner product)** უწოდებენ

MATLAB საშუალებით სკალარული ნამრავლი შეიძლება გამოვთვალოთ შემდეგნაირად:

```
dot_product = sum(A.*B);
```

გავიხსენოთ, რომ A. * B შეცავს ამ ვექტორების შესაბამის წევრთა ნამრავლს. თუ ორივე სტრიქონი ვექტორი, ან ორივე სვეტი ვექტორია, A.*B აგრეთვე ვექტორია, ავიღებთ ამ ვექტორის ელემენტთა ჯამს და მივიღებთ სკალარულ ნამრავლს. თუ A სტრიქონი ვექტორია, ხოლო B სვეტი ვექტორი, მაშინ სკალარული ნამრავლი ორი გზით შეგვიძლია გამოვთვალოთ:

```
dot_product = sum(A'.*B);
dot_product = sum(A.*B');
```

MATLAB–ს აგრეთვე გააჩნია ფუნქცია ვექტორების სკალარული ნამრავლის გამოსათვლელად – **dot.** dot(A,B) იგივე შედეგს მოგვცემს რასაც - sum(A.*B).

4.1.3 მატრიცების გამრავლება

თუ A მატრიცას გავამრავლებთ B მივიღებთ C მატრიცას, რომლის ელემენტებიც A მატრიცის სტრიქონების B მატრიცის სვეტებზე სკალარულ ნამრავლს წარმოადგენს:

$$C(i, j) = \sum_{k=1}^{N} a_{ik} b_{kj}$$

რადგანაც სკალარული ნამრავლი მოითხოვს, რომ ვექტორები ელემენტთა ერთნაირი რაოდენობას შეიცავდნენ, ამითომ როცა მატრიცებს ვამრავლებთ ერთმანეთზე, პირველი მატრიცა (A) თითოეულ სტრიქონში უნდა შეიცავდეს იმდენ წევრს, რამდენსაც შეიცავს მეორე მატრიცის (B) თითოეული სვეტი. ამგვარად, თუ A და B შეცავენ 5 სტრიქონსა და 5 სვეტს, მაშინ მათი გადამრავლების შედეგად მიღებული მატრიცა C უნდა შეიცავდეს 5 სტრიქონსა და 5 სვეტს. ასეთი (კვადრატული მატრიცებისათვის შეგვიძლია გამოვთვალოთ როგორც A*B, ასევე B*A, თუმცა საზოგადოდ ისინი ტოლო არ იქნება.

თუ A შეიცავს 2 სტრიქონსა და 3 სვეტს, და B შეიცავს 3 სტრიქონსა და 3 სვეტს, მათი ნამრავლი C = AB შეიცავს 2 სტრიქონსა და 3 სვეტს:

$$A = \begin{bmatrix} 2 & 5 & 1 \\ 0 & 3 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 2 \\ -1 & 4 & -2 \\ 5 & 2 & 1 \end{bmatrix}$$

პირველი ელემენტი ნამრავლში C = AB იქნება:

$$c_{1,1} = \sum_{k=1}^{3} a_{1k} b_{1k} = a_{1,1} b_{1,1} + a_{1,2} b_{2,1} + a_{1,3} b_{3,1} = 2 \cdot 1 + 5 \cdot (-1) + 1 \cdot 5 = 2$$

ასევე გამოვითვლით სხვა ელემენტებსაც და მივიღებთ მატრიცას:

$$AB = C = \begin{bmatrix} 2 & 22 & -5 \\ -8 & 10 & -7 \end{bmatrix}$$

ჩვენ ვერ გამოვითვლით B*A, ყოველ სტრიქონში არ შეიცავს იმდენსავე ელემენტს, რამდენსაც ყოველ სვეტში.

არსებობს მარტივი წესი იმის დასადგენად, შესაძლებელია თუ არა მოცემული მატრიცების გამრავლება. დავწეროთ ორი მატრიცეს ზომები ერთმანეთის გვერდით. თუ შიდა ორი რიცხვი ტოლია, ასეთი მატრიცების გამრავლება შესაძლებელია და ნამრავლის ზომა განისაზღვრება გარე რიცხვების მიხედვით. მაგალითად განვიხილოთ მოცემული A და B მატრიცების შემთხვევა. A მტრიცის ზომაა 2×3, B – 3×3:

2 X 3 , 3 X 3

შიდა რიცხვები ტოლია და =3, ესე იგი მატრიცების ნამრავლი არსებობს და მისი ზომაა 2×3. B*A შემთხვევაში:

3 × 3 , 2 × 3

შიდა რიცხვები არ არის ტოლი და არ ასებობს ასეთი ნამრავლი.

MATLAB-ში მატრიცების გამრავლება აღინიშნება ნიშნით “ * ”. იმისათვის რომ MATLAB -ში შევქმნათ A და B მატრიცები და შემდეგ A გავამრავლოთ B, გვჭირდება ბრძანებები:

```
A = [2, 5, 1; 0, 3, -1];
B = [1, 0, 2; -1, 4, -2; 5, 2, 1];
C = A*B;
```

თუ MATLAB მივცემთ ბრძანებას B*A, მივიღებთ ინფორმაციას, რომ ასეთი ნამრავლი არ არსებობს.

უმარტივესი გზა ორი სტრიქონი ვექტორის სკალარული ნამრავლის საპოვნელად შემდეგია:

```
dot_product = F*G′
```

თუ დავუშვებთ, რომ მათი სიგრძეა N და ვისარგებლებთ ზემოთ აღწერილი წესით, რომელიც გვაძლევს ნამრავლი მატრიცის ზომას, მივიღებთ:

$$1 \times N, N \times 1$$

F*G ნამრავლი მატრიცის ზომაა 1×1, ანუ ორი ვექტორის სკალარული ნამრავლი სკალარული სიდიდეა. თუ F და G სვეტი ვექტორებია, მათ სკალარულ ნამრავლს ასე გამოვითვლით:

```
dot_product = F′*G
```

შესაძლებელია გამოვითვალოთ ორი ვექტორის გარე ნამრავლი (outer product). დავუშვათ F და G შემდეგი ვექტორებია;

$$F = [2\ \ 5\ \ -1] \qquad G = [0\ \ 1\ \ -3]$$

განვიხილოთ ბრძანება:

```
outer_product = F′*G
```

შევამოწმოთ მათი ზომების თანაფარდობა:

$$3 \times 1, 1 \times 3$$

ნამრავლი არსებობს და შედეგად მიიღება მატრიცა, რომლის ზომაა [3 × 3].
გარე ნამრავლი ისევე განისაზღვრება, როგორც ჩვეულებრივ მატრიცების გამრავლება.
იმისათვის, რომ მატრიცები სწორად გავამრავლოთ, კარგად უნდა დავუკვირდეთ მათ ზომას, და ამის შემდეგ შევარჩიოთ ტრანსპონირების ოპერაციები და თანამამრავლთა სათანადო მიმდევრობა. როგორც ვნახეთ, როცა F და G ვექტორები სტრიქონი ვექტორებია, მაშინ F′ სკალარია, FG′ [3 × 3] ზომის მატრიცა, ხოლო F*G საერთოდ არ არსებობს.

დავუშვათ I კვადრატული ერთეულოვანი მატრიცაა. (გავიხსენოთ მე-3 თავიდან რომ ერთეულოვანი ისეთი მატრიცაა, რომლის მთავარი დიაგონალის ელემენტები ერთის ტოლია, ყველა სხვა ელემენტი კი 0-ის). თუ A იგივე ზომის კვადრატული მატრიცაა, მაშინ AI = IA=A. ეს გამოთვლები გვიჩვენებს, რომ მატრიცის გამრავლებით ერთეულოვან მატრიცაზე იგივე მატრიცას მივიღებთ.

4.1.4 მატრიცის ახარისხება

გავიხსენოთ, რომ თუ A მატრიცაა A.^2 ისეთი ოპერაციაა, რომელსაც ყველა ელემენტი კვადრატში აჰყავს. თუ გვინდა კვადრატში ავიყვანოთ მატრიცა ანუ შევასრულოთ მოქმედება A*A,ვწერთ A^2. A^4 იგივეა, რაც A*A*A*A. როცა ხარისხის მაჩვენებელი არ არის მთელი რიცხვი, უფრო რთულ ოპერაციასთან გვაქვს საქმე. ამ დროს საჭიროა ისეთი სიდიდეების ცოდნა, როგორიცაა მატრიცის მახასიათებელი რიცხვი და მახასითებელი ვექტორი. როგორც ვნახეთ, ორ მატრიცას შორის გამრავლების ოპერაცია რომ შევასრულოთ, პირველი მატრიცის სვეტების რაოდენობა მეორე მატრიცის სტრიქონების რაოდენობას უნდა უდრიდეს, აქედან გამომდინარე, იმისათვის, რომ მატრიცა ავახარისხოთ, იგი აუცილებლად კვადრატული უნდა იყოს.

4.1.5 მატრიცის შებრუნებული

განსაზღვრისათვის, A კვადრატული მატრიცის შებრუნებული არის ისეთი მატრიცა A^{-1}, რომ სრულდება პირობა $AA^{-1} = A^{-1}A$ = ერთეულოვან მატრიცას. მაგალითდ განვიხილოთ ორი მატრიცა A და B:

$$A = \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 1.5 & -5 \\ -2 & 1 \end{bmatrix}$$

თუ გამოვითვლით ნამრავლს A*B და B*A, მივიღებთ შემდეგ მატრიცებს:

$$AB = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad BA = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

ამიტომ A და B ურთიეთშებრუნებული მატრიცებია, ანუ $A = B^{-1}$ და $B = A^{-1}$.

შებრუნებული მატრიცის გამოთვლა მოსაწყენი და დამღლელი პროცესია. საბედნიეროდ MATLAB შეიცავს ფუნქციას **inv**, რომელიც გამოითვლის მატრიცის შებრუნებულს. (არ მოვიყვანთ ალგორითმს მატრიცის შებრუნებულის გამოსათვლელად, იგი განხილული იქნება წრფივ ალგებრასთან დაკავშირებულ თავში. ამგვარად, თუ მივცემთ ბრძანებას inv(A), შედეგად მივიღებთ B მატრიცას და პირიქით.

მატრიცის შებრუნებულის გამოთვლა გვჭირდება მრავალი საინჟინრო ამოცანის გადასაწყვეტად. მოგვიანებით განვიხილავთ ზოგიერთ ასეთ ამოცანას.

სავარჯიშო

MATLA-ის საშუალებით შექმენით შემდეგი მატრიცები და შეასრულეთ მითითებული მოქმედებანი:

$$A = \begin{bmatrix} 2 & 1 \\ 0 & -1 \\ 3 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 3 \\ -1 & 5 \end{bmatrix} \quad C = \begin{bmatrix} 3 & 2 \\ -1 & -2 \\ 0 & 2 \end{bmatrix} \quad D = \begin{bmatrix} 1 & 2 \end{bmatrix} \quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

1. AB

2. DB
3. BC’
4. (CB)D’
5. B^{-1}
6. BB^{-1}
7. $B^{-1}B$
8. AC’
9. $(AC')^{-1}$
10. $(AC')^{-1}(AC')$
11. IB
12. BI

4.1.6 დეტერმინანტი

მატრიცის დეტერმინანტი სკალარია, რომელიც მატრიცის ელემენტების საშუალებით გამოითვლება. დეტერმინანტს ფართო გამოყენება აქვს მრავალი პრობლემის ამოხსნისას, მათ შორის მატრიცის შებრუნებულის გამოთვლის და წრფივ განტოლებათა სისტემის ამოხსნისას. 2×2 ზომის მატრიცის დეტერმინანტი შემდეგნაირად გამოითვლება:

$$|A| = a_{1,1}a_{2,2}-a_{2,1}a_{1,2}$$

მაგალითად , თუ :

$$A=\begin{bmatrix} 1 & 3 \\ -1 & 5 \end{bmatrix}$$

|A| = 8.

3×3 ზომის მატრიცის დეტერმინანტი შემდეგნაირად გამოითვლება:

$$|A| = a_{1,1}a_{2,2}\, a_{3,3}+a_{2,1}a_{2,3}\, a_{3,2}+ a_{1,3}a_{2,1}\, a_{3,2}- a_{3,1}a_{2,2}\, a_{1,3}- a_{3,2}a_{2,3}\, a_{1,1}- a_{3,3}a_{2,1}\, a_{1,2}$$

თუ A შემდეგი მატრიცაა:

$$A=\begin{matrix} 1 & 3 & 0 \\ -1 & 5 & 2 \\ 1 & 2 & 1 \end{matrix}$$

|A| = 5 + 6 + 0 – 0 – 4 – (-3), ანუ 10

უფრო რთულია პროცესი უფრო მეტი ელემენტების შემცველი მატრიცის დეტერმინანტის გამოსათვლელად. არ მოგვყავს საზოგადოდ დეტერმინანტის გამოთვლის პროცესის სრული აღწერა, რადგან MATLAB საშუალებით შეგვიძლია გამოვითვალოთ დეტერმინანტი ფუნქციით **det,** რომლის არგუმენტია კვადრატული მატრიცა.

პრობლემა – ცილის მოლეკულური წონა

გენურ ინჟინერიაში დიდ როლს თამაშობს ისეთი მოწყობილობა, როგორიცაა პროტეინის (ცილის) სინთეზატორი. მას შეუძლია განსაზღვროს ამინომჟავათა რიგი, მიმდევრობა ცილის ჯაჭვისებურ მოლეკულაში. ამინომჟავათა რიგი ეხმარება გენეტიკოსებს დაადგინონ (გააიგივონ) გენი, რომელსაც შეიცავს შექმნილი ცილა. ფერმენტების საშუალებით შესაძლებელია კავშირის დარღვევა მეზობელ გენებს შორის, იმისათვის რომ გამოცალკევდეს საჭირო გენი DNA (დეზოქსირიბონუკლეინის მჟავა)-დან. ეს გენი შემდეგ შეჰყავთ სხვა ორგანიზმში, როგორც ბაქტერია, რომელიც შემდეგ გამრავლდება ახალ გარემოში.

არსებობს მხოლოდ 20 სხვადასხვა ამინომჟავა. ცილის მოლეკულეა შეიცავს ასობოთ ამინომჟავას, რომლებიც დაკავშირებული არიან ერთმანეთთან გარკვეული რიგით. მოცემულ ამოცანაში დავუშვათ რომ დადგენილია პროტეინის მოლეკულაში ამინომჟავათა მომდევრობა და უნდა გამოვთვალოთ ცილის მოლეკულური წონა. ცხრილში მოცემულია ანბანის მიხედვით დალაგებული ამინომჟავათა მწკრივი, მათი მოკლე აღნიშვნა და მოლეკულური წონა.

N	ამინომჟავა	აღნიშვნა	მოლეკულური წონა
1	Alanine	ALA	89
2	Arginine	Arg	175
3	Asparagine	Asn	132
4	Aspartic	Asp	132
5	Cysteine	Cys	121
6	Glutamik	Glu	146
7	Glutamine	Gln	146
8	Glycine	Gly	75
9	Histidine	His	156
10	Isoleucine	Ile	131
11	Leucine	Leu	131
12	Lysine	Lys	147
13	Methionine	Met	149
14	Phenylalanine	Phe	165
15	Proline	Pro	116
16	Serine	Ser	105
17	Threonine	Thr	119
18	Tryptophan	Trp	203
19	Tyrosine	Tyr	181
20	Valine	Val	117

ამ ამოცანის საწყისი მონაცემებია მონაცემთა ფაილი protein.dat, რომელიც შეიცავს ამინომჟავათა რაოდენობას და ტიპს ცილის თითოეულ მოლეკულაში. დავუშვათ მონაცემთა ფაილი შეიქმნა ცილის სინტეზატორის საშუალებით. ფაილის მონაცემთა ყოველი სტრიქონი შეესაბამება ერთ ცილას და შეიცავს 20 მთელ რიცხვს, რომელიც შეესაბამება ცხრილში ამინომჟავას რიგით ნომერს. ამრიგად შემდეგი სტრიქონი:

```
0 0 0 1 0 2 0 0 0 0 0 1 1 0 0 1 0 0 0 0
```

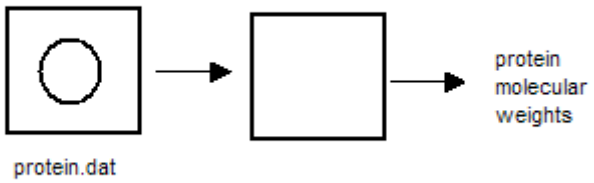
შეესაბამება ცილას ამინომჟავათა მიმდევრობით – LysGluMetAspSerGlu.

1. ამოცანის დასმა

გამოვთვალოთ ცილის მოლეკულური წონა.

2. INPUT/OUTPUT აღწერა

ნახაზზე მოცემული INPUT/OUTPUT დიაგრამა, რომელიც გვიჩვენებს, რომ საწყისი მონაცემები წარმოდგენილია ფაილის სახით, რომელიც შეიცავს მონაცემებს ცილის მოლეკულური შემცველობის შესახებ - რომელი ტიპის ამინომჟავას შეიცავს მოცემული ცილა და რა რაოდენობით.



ნახ. 6.1 INPUT/OUTPUT დიაგრამა

3. სახელდახელო ამოხსნა

დავუშვათ გვაქვს ცილის მოლეკულა LysGluMetAspSerGlu. მათი შესაბამისი მოლეკულური წონებია :

147, 146, 149, 132, 105, 146

აქედან გამომდინარე, ცილის მოლეკულური წონა იქნება 825. მონაცემთა ფაილში ამ ცილას შეესაბამება სტრიქონი:

0 0 0 1 0 2 0 0 0 0 0 1 1 0 0 1 0 0 0 0

ცილის მოლეკულური წონა რომ მივიღოთ ცალკეული ამინომჟავას რაოდენობა უნდა გავამრავლოთ შესაბამის მოლეკულურ წონაზე და მიღებული შედეგები შევკრიბოთ. ასეთი ნამრავლების ჯამი შეგვიძლია განვიხილოთ როგორც ცილის ვექტორის და წონათა ვექტორის სკალარული ნამრავლი. თუ გვინდა გამოვთვალოთ მოლეკულური წონა ცილათა ჯგუფისათვის, შედეგი შეგვიძლია მივიღოთ მატრიცების გადამრავლებით შემდეგნაირად:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 89 \\ 175 \\ 132 \\ 132 \\ 121 \\ 146 \\ 146 \\ 75 \\ 156 \\ 131 \\ 131 \\ 147 \\ 149 \\ 165 \\ 116 \\ 105 \\ 119 \\ 203 \\ 181 \\ 117 \end{bmatrix} = \begin{bmatrix} 825 \\ 1063 \end{bmatrix}$$

4. MATLAB ამოხსნა

MATLAB საშუალებით ეს ამოცანა ძალზე მარტივად ამოიხსნება. ინფორმაცია ამინმჟავების შესახებ წაკითხული იქნება მონაცემთა ფაილიდან და შეიქმნება მატრიცა პროტეინ, განისაზღვრება სვეტი ვექტორი mw, რომლის ელემენტებიც იქნება ანბანის მიხედვით დალაგებულ ამინომჟავათა მოლეკულური წონები. ამ ორი მატრიცის გადამრავლებით მივიღებთ ახალ მატრიცას, რომლის ელემენტებიც იქნება ცილის მოლეკულური წონები.

```
%
%          Tthis  program computes the molecular weights for
%          a group of protein molekules. A data file contains
%          the occurence and number of amino acids in each
%          protein molecule.
%
load protein.dat
mw=[89 175 132 132 121 146 146 75 156 131 131 147 ...
    149 165 116 105 119 203 181 117];
%
%      Compute protein weights
%
weights=protein*mw';
%
%     Print protein weights
```

```
%
[rows cols] = size(protein);
for k=1:rows
    fprintf('protein %3.0f:  molecular weight = %5.0f \n',k,
weights(k))
end
```

5. შემოწმება

დავუშვათ ცილათა ჯგუფი, რომლისთვისაც ვითვლით მოლეკულურ წონებს ასეთია:

```
GlyIleSerThrTrp
AspHisProGln
ThrTyrSerTrpLysMetHisMet
AlaValLeuValMet
LysGluMetAspSerGluLysGluGluGlu
```

მონაცემთა ფაილში გვექნება:

```
0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0 0
0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 1 2 0 0 1 1 1 1 0
1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 2
0 0 0 1 0 4 0 1 0 0 0 2 1 0 0 1 0 0 0 0
```

MATLAB დაგვიბეჭდავს მოლეკულურ წონებს ცილებისათვის:

```
protein   1:  molecular weight =   633
protein   2:  molecular weight =   550
protein   3:  molecular weight =  1209
protein   4:  molecular weight =   603
protein   5:  molecular weight =  1339
```

4.2 მატრიცული მანიპულირება

4.1.7 შემობრუნება

შესაძლებელია მატრიცა შემოვაბრუნოთ 90 გრადუსით საათის ისრის საწინააღმდეგო მიმართულებით ბრძანებით - **rot90**. თუ გვაქვს:

$$A = \begin{bmatrix} 2 & 1 & 0 \\ -2 & 5 & -1 \\ 3 & 4 & 6 \end{bmatrix}$$

თუ გავუშვებთ ბრძანებას :

B = rot90(A); მივიღებთ:

$$B = \begin{bmatrix} 0 & -1 & 6 \\ 1 & 5 & 4 \\ 2 & -2 & 3 \end{bmatrix}$$

ამ ბრძანებას შეიძლება მეორე არგუმენტიც ჰქონდეს, რომელიც განსაზღვრავს რამდენჯერ შემობრუნდეს მატრიცა 90 გრადუსით. ბრძანებები:

```
B = rot90(A);
C = rot90(B);
```

ექვივალენტურია ბრძანების:

```
C = rot90(A,2)
```

flip

მატრიცა შეგვიძლია 'გადავაბრუნოთ' მარჯვნიდან მარცხნივ **fliplr** ან ზემოდან ქვემოთ (ვერტიკალურად) **flipud**:

```
A = [1 2; 4 8; -2 0];
B = fliplr(A);
C = flipud(B);
```

ამ ბრძანებათა საფუძველზე მივიღებთ:

$$A = \begin{bmatrix} 1 & 2 \\ 4 & 8 \\ -2 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 1 \\ 8 & 4 \\ 0 & -2 \end{bmatrix} \qquad C = \begin{bmatrix} -2 & 0 \\ 4 & 8 \\ 1 & 2 \end{bmatrix}$$

4.1.8 მატრიცის ფორმის შეცვლა

ბრძანება **reshape** მოცემულ მატრიცას შეუცვლის ფორმას – თანაფარდობას სტრიქონების და სვეტების რაოდენობას შორის. ფუნქციის არგუმენტები ისე უნდა შეირჩეს, რომ საწყის და შედეგად მიღებულ მატრიცების ელემენტთა რაოდენობა ერთნაირი იყოს. ამ ფუნქციას გააჩნია სამი არგუმენტი. პირველი, თავად მატრიცაა, ბოლო ორი კი განსაზღვრავს ახალი მატრიცის სტრიქონების და სვეტების რაოდენობას. მაგალითად განვიხილოთ შემდეგი ბრძანებები:

```
A = [2 5 6 –1; 3 –2 10 0];
B = reshape(A,4,2);
C = reshape(A,8,1);
```

მივიღებთ:

$$A=\begin{bmatrix}2 & 5 & 6 & -1\\ 3 & -2 & 10 & 0\end{bmatrix} \qquad B=\begin{bmatrix}2 & 6\\ 3 & 10\\ 5 & -1\\ -2 & 0\end{bmatrix} \qquad C=\begin{bmatrix}2\\ 3\\ 5\\ -2\\ 6\\ 10\\ -1\\ 0\end{bmatrix}$$

4.1.9 მატრიცის ნაწილის ამოღება ახალი მატრიცის სახით

ფუნქციები **diag, triu, tril** საშუალებას გვაძლევს მატრიცის ელემენტები ამოვიღოთ. სამივე ბრძანებაში იგულისხმება მატრიცის მთავარი დიაგონალი, თუ იგი კვადრატული არაა. მთავარი იმ დიაგონალს ეწოდება, რომელიც იწყება ზედა მარცხენა კუთხიდან და ელემენტების სვეტის და სტრიქონის მიმთითებელი ინდექსები ერთნაირია – $a_{1,1}$, $a_{2,2}$ და ა.შ. მთავარი დიაგონალი აქვთ როგორც კვადრატულ, ასევე არაკვადრატულ მატრიცებს. მაგალითად A მატრიცის მთავარი დიაგონალის ელემენტებია 2, -2. B – 2, 10, ხოლო C – 2. ფუნქცია **diag(A)** შექმნის სვეტ ვექტორს, რომლის ელემენტებიც A მატრიცის მთავარი დიაგონალის ელემენტებს შეიცავს.

შეიძლება ამ ფუნქციას მეორე არგუმენტიც ჰქონდეს **diag(A,k).** იმ შემთხვევაში თუ გვსურს დიაგონალის რიგი მივუთითოთ. თუ k > 0, მთავარი დიაგონალის ზემოთ k-ურ დიაგონალი შეირჩევა, თუ k <0, მთავარი დიაგონალის ქვემოთ k-ური დიაგონალი იქნება შერჩეული.

თუ **diag** ფუნქციის არგუმენტად ნაცვლად მატრიცისა შერჩეულია ვექტორი, მაშინ ფუნქცია შექმნის კვადრატულ მატრიცას, რომლის მთავარი დიაგონალც მოცემული ვექტორის ელემენტებია, ყველა სხვა ელემენტი კი 0-ის ტოლია. მაგალითად:

```
V = [1 2 3];
A = diag(v);
mogvcems:
```

$$A=\begin{bmatrix}1 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 3\end{bmatrix}$$

ფუნქცია **triu(A)** შექმნის მატრიცას, რომელიც შეიცავს A მატრიცის მთავარი დიაგონალის და მის ზემოთ განლაგებულ ელემენტებს, დანარჩენი ელემენტები ნულის ტოლია. ამ ფუნქციას შეიძლება მეორე არგუმენტიც ჰქონდეს. ფუნქცია **triu(A,k)** მოგვცემს მატრიცას, რომელიც იგივე ზომისა, რაც A , შეიცავს მის ელემენტებს k-ური დიაგონალს ზემოთ, ან ქვემოთ, სხვა ყველა ელემენტი 0-ის ტოლია. განვიხილოთ MATLAB ბრძანებები:

```
A = [1:2:7; 3:3:12; 4:-1:1; 1:4];
B = triu(A);
C = triu(A,-1);
D = triu(A,3);
```

შედეგად მიიღება მატრიცები:

$$A=\begin{bmatrix}1&3&5&7\\3&6&9&12\\4&3&2&1\\1&2&3&4\end{bmatrix} \qquad B=\begin{bmatrix}1&3&5&7\\0&6&9&12\\0&0&2&1\\0&0&0&1\end{bmatrix}$$

$$C=\begin{bmatrix}1&3&5&7\\3&6&9&12\\0&3&2&1\\0&0&3&4\end{bmatrix} \qquad D=\begin{bmatrix}0&0&0&7\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}$$

Tril ფუნქცია მსგავსია **triu** ფუნქციისა, მხოლოდ ის ქმნის ქვედა სამკუთხა მატრიცას. თუ წინა მაგალითში შევცვლით **triu** ფუნქციას **tril** –ით, მივიღებთ:

$$A=\begin{bmatrix}1&3&5&7\\3&6&9&12\\4&3&2&1\\1&2&3&4\end{bmatrix} \qquad B=\begin{bmatrix}1&0&0&0\\3&6&0&0\\4&3&2&0\\1&2&3&4\end{bmatrix}$$

$$C=\begin{bmatrix}0&0&0&0\\3&0&0&0\\4&3&0&0\\1&2&3&0\end{bmatrix} \qquad D=\begin{bmatrix}1&3&5&7\\3&6&9&12\\4&3&2&1\\1&2&3&4\end{bmatrix}$$

სავარჯიშო

განსაზღვრეთ მატრიცები, რომელიც შეიქმნება შემდეგი ფუნქციების მოქმედების შედეგად, თუ ვიცით, რომ:

$$A=\begin{bmatrix}0&-1&0&3\\4&3&5&0\\1&2&3&0\end{bmatrix} \qquad B=\begin{bmatrix}1&3&5&0\\3&6&9&12\\4&3&2&1\\1&2&3&4\end{bmatrix}$$

1. rot90(B)
2. rot90(A,3)
3. fliplr(A)
4. flipud(fliplr(B))
5. reshape(A,4,3)
6. reshape(A,6,2)
7. reshape(A,2,6)

8. reshape(flipud(B),8,2)
9. triu(B)
10. triu(B,-1)
11. tril(A,2)
12. diag(rot90(B))

გამონასახთა შეთავსება (image alignment)

ციფრული გამონასახი წარმოდგენილია მატრიცის სახით, რომლის ელემენტებიც სინათლის ინტენსივობას შეესაბამება. ასეთი მატრიცის ელემენტებს პიქსელებს ანუ სურათის ელემენტებს უწოდებენ. მაღალი გარჩევის გამონასახი შეიცავს ელემენტთა დიდ რაოდენობას, დაბალი გარჩევის – მცირე რაოდენობას. მაგალითად მაღალი გარჩევის გამონასახი შესაძლოა შეიცავდეს 1024 სტრიქონს და ამდენივე სვეტს, ესე იგი პიქსელების საერთო რაოდენობა მილიონზე მეტი იქნება. თითოეული სიდიდე ასეთ მატრიცაში არის კოდი, რომელიც სინათლის ინტენსივობას შეესაბამება. კოდი შეიძლება შეიცავდეს ინფორმაციას ფერის ან შავ-თეთრი გამონასახის შემთხვევაში ნაცრისფრის სხვადასხვა ტონალობების შესახებ.

დავუშვათ გამონასახი წარმოდგენილია 6 სტრიქონიანი და 6 სვეტიანი მატრიცის სახით. ასევე დავუშვათ, რომ მატრიცის თითოეული ელემენტი 0 და 7 შორისაა მოთავსებული, რაც რუხი ფერის ტონალობებს ეთანადება. მაგალითად:

$$\begin{bmatrix} 0 & 0 & 2 & 6 & 2 & 0 \\ 0 & 1 & 0 & 6 & 6 & 0 \\ 1 & 0 & 0 & 2 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

დავუშვათ გვაქვს ერთიდაიგივე ობიექტის ორი გამონასახი ეთიდაიგივე გარჩევით და რუხი ფერის ტონალობათა კოდით. ასევე დავუშვათ, რომ არ ვიცით გამონასახებს ერთმანეთის მიმართ როგორი მდებარეობა უკავიათ. იმისათვის, რომ ისინი ერთმანეთს შევუთავსოთ, ერთერთი მათგანი უცვლელად დავტოვოთ, მეორე კი მატრიცული მანიპულაციებით შევუთავსოთ მას. ისინი შეთავსებულად ჩათვლება, როცა შესაბამისი ელემენტების მნიშვნელობები ერთმანეთს დაემთხვევა. მაგალითად დავუშვათ A და B მატრიცები ერთიდაიგივე ობიექტის გამონასახებია:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 2 & 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

იმისათვის, რომ B შევუთავსოთ A, იგი უნდა შემოვაბრუნოთ 270 გრადუსით საათის ისრის საწინააღმდეგოდ (ან 90 გრადუსით საათის ისრის მიმართულებით). ანდა გადავაბრუნოთ

ქვემოდან ზემოთ და შემოვაბრუნოთ 90 გრადუსით საათის ისრის საწინააღმდეგოდ. შეამოწმეთ სამივე გზა, რათა დარწმუნდეთ, რომ გამონასახები ამგვარად შეთავსდებიან.

იმისათვის, რომ განვსაზღვროთ შეუთავსდა თუ არა ორი გამონასახი (image 1 da image 2) ერთმანეთს, შეგვიძლია გამოვთვალოთ სხვაობები შესაბამის ელემენტებს შორის და მიღებული შედეგები შევკრიბოთ. ამას მივაღწევთ MATLAB ბრძანებით:

```
dif = sum(sum(image 1 – image 2));
```

sum ფუნქცია ორჯერ გავიმეორეთ იმიტომ, რომ ყველა სხვაობა შეგვეკრიბა. პირველი **sum** მოგვცემს ვექტორს, რომლის ელემენტებია შესაბამისი სვეტების ელემენტების ჯამი, ხოლო მეორე **sum** შეკრებს ამ ვექტორის ელემენტებს. სამწუხაროდ, შესაძლოა ეს ჯამი 0 გამოვიდეს. განვიხილოთ ორი გამონასახის შესაბამისი მატრიცა:

$$\begin{bmatrix} 1 & 7 \\ 5 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix}$$

თუ გამოვითვლით ამ ორი მატრიცის შესაბამისი ელემენტებს შორის სხვაობების ჯამს მივიღებთ 5 + (-5) = 0. თუმცა ცხადად ჩანს, რომ ისინი ართიდაიგივე გამონასახები ნამდვილად არ არის. 0 იმიტომ მივიღეთ, რომ დადებითმა და უარყოფითმა სხვაობებმა გააბათილა ერთმანეთი. თუ სხვაობებს კვადრატში ავიყვანთ, ან მათ აბსოლუტურ სიდიდეს ავიღებთ და ისე დავითვლით ჯამს, ეს სიდიდეები ერთმანეთს ვეღარ გააბათილებს. MATLAB ბრძანებით მიიღება ასე გამოთვლილი სხვაობათა ჯამი, რომელსაც ვუწოდებთ მანძილის ზომას:

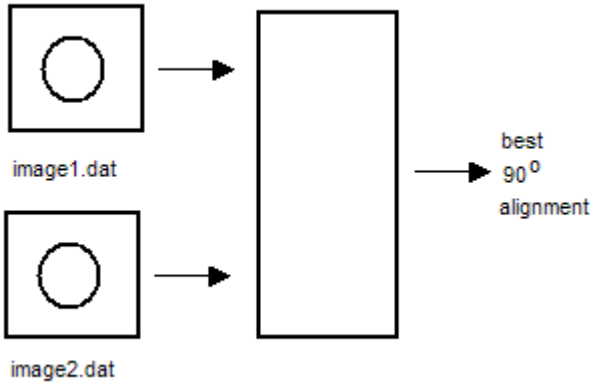
```
distance = sum(sum(image1 – image2).^2));
```

შეგვიძლია დავითვალოთ ეს მანძილები ყველა შესაძლო შეთავსების შემთხვევაში. ორი გამონასახი ჩაითვლება შეთავსებულად, თუ მანძილი 0 ტოლია. თუ გავითვალისწინებთ, რომ ერთიდაიმავე ობიექტის ორი სხვადასხვა გამონასახის შესაბამის ელემენტებს შეიძლება მცირედ განსხვავებული მნიშვნელობები ჰქონდეს (გამოწვეული ინსტრუმენტული ცთომილებით ან საკომუნიკაციო არხებში ხმაურით), შეგვიძლია გამოვთვალოთ მანძილები ყველა შესაძლო შეთავსებისთვის და შემდეგ ავირჩიოთ მათ შორის უმცირესი.

1. ამოცანის დასმა

განვსაზღვროთ ორი გამონასახის შეთავსებისათვის რა სახის მანიპულაციებია საჭირო.

2. INPUT/OUTPUT აღწერა



ნახ. 6.2 INPUT/OUTPUT დიაგრამა

ნახ. 6.2 წარმოადგენს INPUT/OUTPUT დიაგრამას, სადაც ნაჩვენებია, რომ საწყის მონაცემებს ვიღებთ ორი ფაილიდან, შედეგი კი წარმოადგენს სიდიდეს: ერთერთი გამონასახის 90 გრადუსით რამდენჯერ შებრუნება დაგვჭირდა, რომ იგი მეორე გამონასახს შეთავსებოდა.

3. სახელდახელო ამოხსნა

დავუშვათ გვაქვს ორი გამონასახი:

$$C = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix} \qquad D = \begin{bmatrix} 1 & 3 \\ 3 & 4 \end{bmatrix}$$

თუ შევაბრუნებთ მიმდევრობით 0°, 90°, 180°, 270°, მივიღებთ:

$$\begin{bmatrix} 1 & 3 \\ 3 & 4 \end{bmatrix} \quad \begin{bmatrix} 3 & 4 \\ 1 & 3 \end{bmatrix} \quad \begin{bmatrix} 4 & 3 \\ 3 & 1 \end{bmatrix} \quad \begin{bmatrix} 3 & 3 \\ 4 & 1 \end{bmatrix}$$

ახლა თუ გამოვითვლით მანძილებს (ელემენტებს შორის სხვაობათა კვადრატების ჯამს) C მატრიცასა და D –ს ამ ოთხ ვერსიას შორის, მივიღებთ: 19, 7, 1 და 13 შესაბამისად. როგორც ვხედავთ მინიმალური მანძილი = 1, რასაც შეესაბამება საათის ისრის საწინააღმდეგოდ 180 გრადუსით შემობრუნება.

4. MATLAB ამოხსნა

ვუშვებთ, რომ ორი გამონასახი ჩაწერილია ASCII მონაცემთა ფაილის სახით. საჭიროა 4 ციკლის გამოთვლა, რომ მივიღოთ მობრუნების 4 სხვადასხვა მნიშვნელობა. შემდეგ ვსარგებლობთ **min** ფუნქციით, რომ შევარჩიოთ მინიმალური მანძილი და მისი შესაბამისი მდებარეობა მანძილების ვექტორში. ასე განვსაზღვრავთ თუ რამდენჯერ შემოვაბრუნეთ გამონასახი შეთავსების მისაღწევად.

```
%
%     This program determines the best alignment between
%     two images using rotation of 90 degree
%
%     load image1.dat
%     load image2.dat
%
%     Compute rotational distances.
%
for k=0:3
    a=rot90(image2,k);
    distance(k+1)=sum(sum(image1-a).^2);
end
%
%     Print best alignment
%
[minval, minloc]=min(distance);
fprintf('Image alignment best at %3.0f degrees \n',...
    (minloc-1)*90)
fprintf('(counterclockwise) \n \n')
```

5. შემოწმება

შევნიშვავთ, რომ ეს პროგრამა იმუშავებს ნებისმიერი გარჩევის გამონასახებისათვის. ერთადერთი მოთხოვნაა, რომ გამონასახების შესაბამის მატრიცებს ერთნაირი ზომა ჰქონდეთ. თუ ამ პროგრამას შევამოწმებთ ზემოთ განხილული A და B მატრიცებისათვის მივიღებთ:

```
Image alignment best at 270 degrees
(counterclockwise)
```

ამ თავში შევაჯამეთ მატრიცული გამოთვლების და მანიპულაციების ოპერაციები. განვსაზღვრეთ მატრიცის ტრანსპონირებული და შებრუნებული. ვნახეთ როგორ გამოვთვალოთ ორი ვექტორის სკალარული ნამრავლი და როგორ გადავამრავლოთ ერთი მატრიცა მეორეზე. გავეცანით MATLAB ფუნქციებს, რომელთა საშუალებითაც შეგვიძლია შევცვალოთ მატრიცის ფორმა და სტრუქტურა. ფუნქციით **rot90** შეგვიძლია შემოვაბრუნოთ მატრიცის ელემენტები საათის ისრის საწინარმდეგო მომართულებით. **reshape** ფუნქცია საშუალებას გვაძლევს შევქმნათ ახალი მატრიცა ელემენტების იგივე რაოდენობით. გავეცანით ფუნქციებს, რომელთა საშუალებით შეგვიძლია მატრიციდან ამოვიღოთ ელემენტები და ასე შევქმნათ ახალი მატრიცა ან ვექტორი.

სპეცსიმბოლოები

'	განსაზღვრავს მატრიცის ტრანსპონირებას
*	მატრიცების გამრავლება

ბრძანებები და ფუნქციები

diag	ამოიღებს მატრიცის მთავარი დიაგონალის ელემენტებს
det	გამოითვლის მატრიცის დეტერმინანტს
fliplr	გადააბრუნებს მატრიცას მარცხნიდან მარჯვნივ (ჰორიზონტალურად)
flipud	გადაბრუნებს მატრიცას ზემოდან ქვემოთ (ვერტიკალურად)
inv	გამოითვლის მატრიცის შებრუნებულს
reshape	ფორმას უცვლის მატრიცას
rot90	შემოაბრუნებს მატრიცას 90 გრადუსით საათის ისრის საწინააღმდეგოდ
tril	შექმნის ქვედა სამკუთხა მატრიცას
triu	შექმნის ზედა სამკუთხა მატრიცას
dot	გვაძლევს ორი ვექტორის სკალარულ ნამრავლს

პრობლემები

პრობლემები 1 - 10 დაკავშირებულია ამ თავში განხილულ ამოცანებთან, ხოლო 11 – 21 უკავშირდება სხვა საინჟინრო ამოცანებს.

ამოცანები

ცილის მოლეკულური წონები. ეს ამოცანები უკავშირდება ამ თავში განხილულ ამოცანას ცილის მოლეკულური წონის განსაზღვრის თაობაზე.

1. შეცვალე პროგრამა ისე, რომ გამოითვალოს და დაიბეჭდოს ცილის რიგითი ნომერი და მოლეკულური წონა იმ ცილისათვის, რომელსაც უდიდესი მოლეკულური წონა გააჩნია.
2. შეცვალე პროგრამა ისე, რომ დაიბეჭდოს განხილული ჯგუფის მიხედვით ცილის საშუალო მოლეკულური წონა
3. შეცვალე პროგრამა ისე, რომ დაგვიბეჭდოს ფაილში ჩაწერილ მონაცემებში ცალკეული ამინომჟავა სულ რამდენჯერ გვხვდება. დაიბეჭდოს შემდეგი ფორმატით

	ამინომჟავას ხდომილების შეჯამება
ამინომჟავას ნომერი	ხდომილების რაოდენობა
1	XXX
2	XXX
. . .	
20	XXX

4. შეცვალე პროგრამა ისე, რომ დაბეჭდოს იმ ცილის რიგითი ნომერი და მოლეკულური წონა, რომელიც ამინომჟავათა ყველაზე მეტ სახეობას შეიცავს.

5. შეცვალე პროგრამა ისე, რომ დაბეჭდოს არსებული მონაცემების საფუძველზე საშუალოდ ამინომჟავას რამდენ ატომს შეიცავს ცალკეული ცილის მოლეკულა.

გამონასახთა შეთავსება. ეს ამოცანებუ უკავშირდება ამ თავში განხილულ ამოცანას ერთიდაიგივე ობიექტის ორი გამონასახის შეთავსების თაობაზე.

6. შეცვალე პროგრამა ისე, რომ დაბეჭდოს აგრეთვე გამოთვლილი მანძილები გამონასახის ყოველი შემობრუნების შემდეგ.
7. შეცვალე პროგრამა ისე, რომ დაბეჭდოს შემობრუნების კუთხე გრადუსებში საათის ისრის მიმართულებით.
8. შეცვალე პროგრამა ისე, რომ შეთავსებისას გამიყენოს MATLAB ფუნქციები fliplr და flipud.
9. შეცვალე პროგრამა ისე, რომ შეადაროს გამონასახები მეორე გამონასახის გადაბრუნებით მარცხნიდან მარჯვნივ (ჰორიზონტალურად) და ასევე საათის ისრის მიმართულებით 90, 180 და 270 გრადუსით შემობრუნებული გამონასახის გადაბრუნებით. (მოიფიქრეთ, რატომ არ ჩავრთეთ დამატებით შემთხვევა ზემოდან ქვემოთ (ვერტიკალურად) გადაბრუნება?)
10. შეცვალე პროგრამა ისე, რომ მანძილი გამოთვალოს როგორც შესაბამის ელემენტებს შორის სხვაობათა აბსოლუტური სიდიდეების ჯამი. შეადარეთ ორივე შემთხვევაში მიღებული შედეგები.

ამინომჟავები. ცილის მოლეკულის შემადგენელი ამინომჟავები შეიცავს შემდეგ ქიმიურ ელემენტებს: ჟანგბადი(O), ნახშირბადი(C), აზოტი(N), გოგირდი(S) და წყალბადი(H), როგორც ნაჩვენებია ცხრილში დავუშვათ ამ ცხრილის მონაცემები ჩაწერილია ფაილში elements.dat. ამ ელემენტთა ატომური წონებია:

Oxigen	15.9994
Carbon	12.011
Nitrogen	14.00674
Sulfur	32.066
Hydrogen	1.00794

11. დაწერეთ პროგრამა, რომელიც გამოითვლის თითოეული ამინომჟავას მოლეკულურ წონას და შექმნის მონაცემთა ფაილს aaweights.dat, რომელიც შეიცავს მონაცემებს ფაილიდან elements.dat პლუს ამინომჟავას მოლეკულური წონა.

ამინომჟავათა მოლეკულები

ამინომჟავა	O	C	N	S	H
Alanine	2	3	1	0	7
Arginine	2	6	4	0	15
Asparagine	3	4	2	0	8
Aspartic	4	4	1	0	6
Cysteine	2	3	1	1	7
Glutamik	4	5	1	0	8
Glutamine	3	5	2	0	10
Glycine	2	2	1	0	5
Histidine	2	6	3	0	10

Isoleucine	2	6	1	0	13
Leucine	2	6	1	0	13
Lysine	2	6	2	0	15
Methionine	2	5	1	1	11
Phenylalanine	2	9	1	0	11
Proline	2	5	1	0	10
Serine	3	3	1	0	7
Threonine	3	4	1	0	9
Tryptophan	2	11	2	0	11
Tyrosine	3	9	1	0	11
Valine	2	5	1	0	11

12. 11 ამოცანისათვის დაწერილი პეოგრამა შეცვალე ისე, რომ გამოთვალოს და დაბეჭდოს ამინომჟავავთა საშუალო მოლეკულური წონა.
13. 11 ამოცანისათვის დაწერილი პეოგრამა შეცვალე ისე, რომ გამოთვალოს და დაბეჭდოს იმ ამინომჟავას რიგითი ნომერი რომელსაც აქვს უდიდესი და უმცირესი მოლეკულური წონა.

მატრიცული ანალიზი. შექმენით მატრიცა ამოცანის პირობის გათვალისწინებით და შეინახეთ იგი ASCII ფაილში array.dat, რომელსაც შემდეგ წაიკითხავს პროგრამა და გააანალიზებს.

14. დაწერეთ პროგრამა, რომელიც წაიკითხავს მატრიცას ფაილიდან array.dat, განსაზღვრავს არის თუ არა იგი ზედა სამკუთხა მატრიცა და დაბეჭდავს შესაბამისად: “Upper Triangular” ან “Not Upper Triangular”
15. დაწერეთ პროგრამა, რომელიც წაიკითხავს მატრიცას ფაილიდან array.dat, განსაზღვრავს არის თუ არა იგი ქვედა სამკუთხა მატრიცა და დაბეჭდავს შესაბამისად: “Lower Triangular” ან “Not Lower Triangular”
16. დაწერეთ პროგრამა, რომელიც წაიკითხავს მატრიცას ფაილიდან array.dat, განსაზღვრავს არის თუ არა იგი დიაგონალური მატრიცა და დაბეჭდავს შესაბამისად: “Diagonal” ან “Not Diagonal”. თუ დიაგონალური მატრიცა აგრეთვე ერთეულოვანიცაა, დაბეჭდავს “Identity” “Diagonal” ნაცვლად.
17. სიმეტრიული ეწოდება კვადრატულ მატრიცას, რომელიც სიმეტრიულია მთავარი დიაგონალის მიმართ. ასეთი მატრიცას ტრანსპონირებული იგივე მატრიცის ტოლია. დაწერეთ პროგრამა, რომელიც წაიკითხვს მატრიცას ფაილიდან array.dat, განსაზღვრავს არის თუ არა იგი სიმეტრიული და დაბეჭდავს შესაბამისად: “Symmetric” ან “Not Symmetric”
18. ტოეპლიცის (Toeplitz) მატრიცა ეწოდება ისეთ მატრიცას, რომლის დიაგონალის ელემენტები ერთმანეთის ტოლია, მაგრამ სხვადასხვა დიგონალის ელემენტები განსხვავდება. დაწერეთ პროგრამა, რომელიც წაიკითხვს მატრიცას ფაილიდან array.dat, განსაზღვრავს არის თუ არა იგი ტოეპლიცის (Toeplitz) და დაბეჭდავს შესაბამისად: “Toeplitz” ან “Not Toeplitz”.
19. ტრიდიაგონალური მატრიცა ეწოდება ისეთ მატრიცას, რომლის მხოლოდ მთავარი დიაგონალის, მთავარი დიაგონალის ზედა და ქვედა ორი დიაგონალის ელემენტებია არანულოვანი. დაწერეთ პროგრამა, რომელიც წაიკითხვს მატრიცას ფაილიდან array.dat, განსაზღვრავს არის თუ არა იგი ტრიდიაგონალური და დაბეჭდავს შესაბამისად: “Tridiagonal” ან “Not Tridiagonal”.

20. ზოგიერთი რიცხვითი მეთოდი საჭიროებს მატრიცის სტრიქონების ისეთ გადალაგებას, როცა საჭიროა პირველ სტრიქონად გადავიდეს ის ატრიქონი, რომელიც შეცავს პირველი სვეტის ელემენტებს შორის აბსოლუტური სიდიდით უდიდეს. შემდეგ თუ განვიხილავთ დარჩენილ სტრიქონებს მეორე სტრიქონად გადავა ის სტრიქონი, რომელიც შეცავს მე-2 სვეტის ელემენტებს შორის აბსოლუტური სიდიდით უდიდეს. პროცესი გრძელდება, ვიდრე ამ წესით არ დალაგდება მატრიცა მთლიანად. ამ პროცესს უწოდებენ (row pivoting). დაწერეთ ასეთი პროგრამა და დაალაგეთ თქვენს მიერ შექმნილი 10 სტრიქონიანი მატრიცა.
21. Column pivoting ითვალიცწინებს მონაცემთა იმის მსგავს გადალაგებას, როგოც row pivoting, მხოლო ამ შემთხვევაში სვეტების გადალაგება ხდება: პირველი სვეტი შეიცავს პირველი სტრიქონის აბსოლუტური სიდიდით უდიდეს მნიშვნელობას და ა.შ. დაწერეთ შესაბამისი პროგრამა.

დედამიწა კოსმოსიდან (გადაღებულია Apollo 17 კოსმოსური ხომალდიდან)

4 MATLAB-ის გრაფიკული შესაძლებლობები

პრობლემა: გლობალური ცვლილების პროგნოზი

ბიოსფერო ის გარემოა, სადაც სიცოცხლე არსებობს. დედამიწის ბიოსფერო მოიცავს ატმოსფეროს, ჰიდროსფეროს – ზღვებსა და ოკეანეებს და ლითოსფეროს – დედამიწის ქერქის ნაწილს. იმისათვის რომ შევისწავლოთ ბიოსფეროში მიმდინარე ფიზიკური პროცესები, კარგად უნდა გავერკვეთ უამრავ წვრილმან დეტალში, როგორიცა მაგალითად, ნახშირორჟანგის მიმოქცევა ატმოსფეროსა და ოკეანეში, ოზონის საფარის შესუსტება, ქიმიური და ენერგეტიკული პროცესებით გამოწვეული კლიმატური ცვლილებები. საჭირო მინაცემების შესაგროვებლად გამოიყენება მეტეოროლოგიური რაკეტა, რომლის საშუალებით ხერხდება დედამიწის ატმოსფროს ზედა ფენების კვლევა. ატმოსფეროს სხადასხვა ფენის გავლისას რაკეტის ცხვირზე დამაგრებული ტელემეტრული სისტემა გადმოსცემს მონაცემებს დედამიწაზე. დაგროვილი მონაცემები შემდგომში კომპიუტერის საშუალებით მუშავდება.

შესავალი

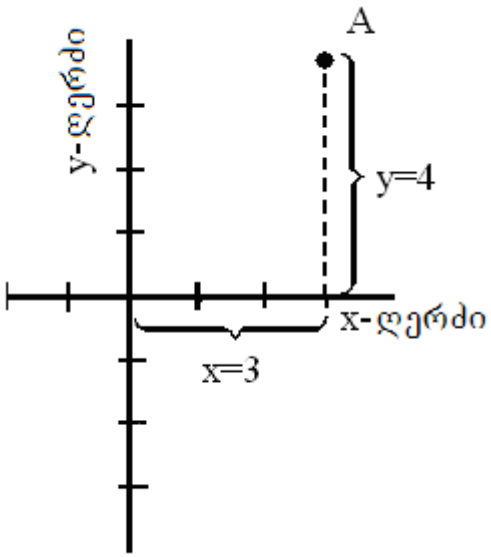
გრაფიკის აგება საშუალებას გვაძლევს შევამოწმოთ და გავაანალიზოთ მონაცემები. MATLAB -ს გაჩნია მძლავრი გრაფიკული საშუალებები. თუ შეისწავლით მათ, საშუალება გექნებათ მარტივად ააგოთ სხვადასხვა ფორმატის გრაფიკი, როგორიცაა x – y გრაფიკი, პოლარული, Bar და კონტურული, სამგანზომილებიანი გრაფიკი. ამ თავში განვიხილავთ MATLAB –ის გრაფიკულ ბრძანებებს და მათ მრავალფეროვან შესაძლებლობებს

4.1 X –Y გრაფიკი

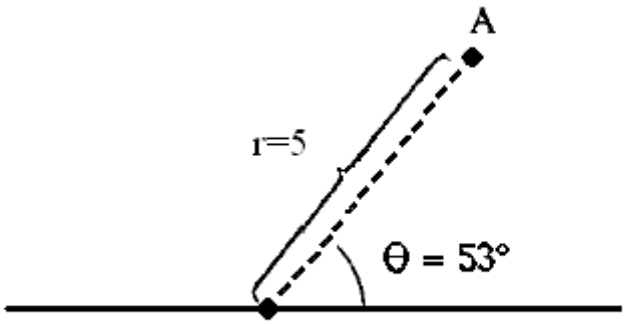
განვიხილოთ X – Y გრაფიკი, რომელიც ყველაზე ხშირად გამოიყენება. მონაცემები, რომელთა მიხედვითაც გრაფიკს ვაგებთ, ჩვეულბრივ მონაცემთა ფაილის სახითაა წარმოდგენილი ან მიიღება კომპიუტერული პროგრამით გათვალისწინებული გამოთვლებით, რომელიც იძლევა შესაბამის x და y ვექტორებს. საზოგადოდ ვთვლით, რომ x შეესაბამება დამოუკიდებელ ცვლადს, ხოლო y მასზე დამოკიდებულ ცვლადს. y მნიშვნელობები შესაძლოა გამოთვლილი იქნას როგორც x –ის ფუნქცია. შესაძლოა ორივე სიდიდე მიღებული იყოს ექსპერიმენტის შედეგად.

კოორდინატთა მართკუთხა სისტემა (დეკარტის)

სხვადასხვა ამოცანებში ვიყენებთ მონაცემებს რომელთა წარმოდგენა შესაძლებელია როგორც კოორდინატთა მართკუთხა, ისე პოლარულ სისტემაში. კოორდინატთა მართკუთხა სისტემაში წერტილის ადგილი განისაზღვრება მანძილით სათავიდან ჰორიზონტალური და ვერტიკალური ღერძების გასწვრივ ნახ 7.1. გრაფიკის ასაგებად კოორდინატთა პოლარულ სისტემაში გვჭირდება პოლარული მონაცემები – მანძილი კოორდინატთა სათავიდან და კუთხე ნახ 7.2. ჯერ განვიხილოთ გრაფიკის აგება კოორდინატთა მართკუთხა, დეკარტის სისტემაში, შემდეგ პოლარულ სისტემაში. ვიდრე წარმოგიდგენთ ბრძანებებს გრაფიკის ასაგებად, მიმოვიხილოთ უკვე განხილული ბრძანებები რომლებიც ნახაზს უკეთებს სათაურს და ღერძებს შესაბამის წარწერას დაურთავს. გრაფიკს დაუსრულებელი სახე აქვს თუ არ შეიცავს ამგვარ ინფორმაციას. ხშირად გრაფიკს დავურთავთ საკოორდინატო ბადეს, რადგან იგი აადვილებს იმ სიდიდეთა შეფასებას, რომლის მიხედვითაც იგი აიგება.



ნახ 7.1. კოორდინატთა მართკუთხა - დეკარტის სისტემა



ნახ 7.2. კოორდინატთა პოლარული სისტემა

Labels (წარწერა ღერძებზე)

title('text')	გრაფიკს გაუკეთებს სათაურს
xlabel('text')	x-ღერძის ქვემოთ წააწერს ბრჭყალებში მოთავსებულ ტექსტს
ylabel	y-ღერძის გასწვრივ წააწერს ბრჭყალებში მოთავსებულ ტექსტს
text(x,y,'text')	ბრჭყალებში ჩაწერილ ტექსტს წააწერს გრაფიკს წერტილში, რომლის კოორდინატებია x,y მოცემული გრაფიკის ღერძების მიხედვით, თუ x ,y ვექტორებია, წარწერას გააკეთებს ყოველ წერტილში
text(x,y, 'text','sc')	ბრჭყალებში ჩაწერილ ტექსტს წააწერს გრაფიკს წერტილში, რომლის კოორდინატებია x,y ისე, რომ ქვედა მარცხენა კუთხის კოორდინატებია (0,0), ზედა მარჯვენასი კი (1,1)
gtext('text')	წააწერს ტექსტს გრაფიკზე იმ წერტილში, რომელსაც მაუზით ან კლავიატურის ისრებით მივუთითებთ
grid	დაიტანს გრაფიკზე საკოორდინატო ბადეს

4.1.1 ბრძანება plot

სზოგადოდ როცა ამ ბრძანებით ვაგებთ გრაფიკს, იგულისხმება, რომ x და y ღერძები დაყოფილია თანატოლ ინტრვალებად. ასეთ გრაფიკულ ოპერაციას წრფივს უწოდებენ. ზოგჯერ შეიძლება საჭირო იყოს ლოგარითმული გრადაცია ერთ-ერთ ან ორივე ღერძზე. 10 ფუძიანი ლოგარითმული გრადაცია მოხერხებულია, როცა ცვლადის მნიშვნელობები ძალიან ფართო დიაპაზონშია განსაზღვრული.
MATLAB ბრძანებები წრფივი და ლოგარითმული გრადაციის გრაფიკებისათვის შემდეგია:

plot(x,y)	აგებს x,y სიდიდების წრფივ გრაფიკს x – დამოუკიდებელი ცვლადია, y მასზე დამოკიებული
semilogx(x,y)	აგებს გრაფიკს ლოგარითმული გრადაციით x –თვის და წრფივით y –თვის.
semilogy(,y)	აგებს გრაფიკს ლოგარითმული გრადაციით y –თვის და წრფივით x –თვის.
loglog(x,y)	აგებს x,y გრაფიკს ლოგარითმული გრადაციით ორივე

ცვლადიათვის

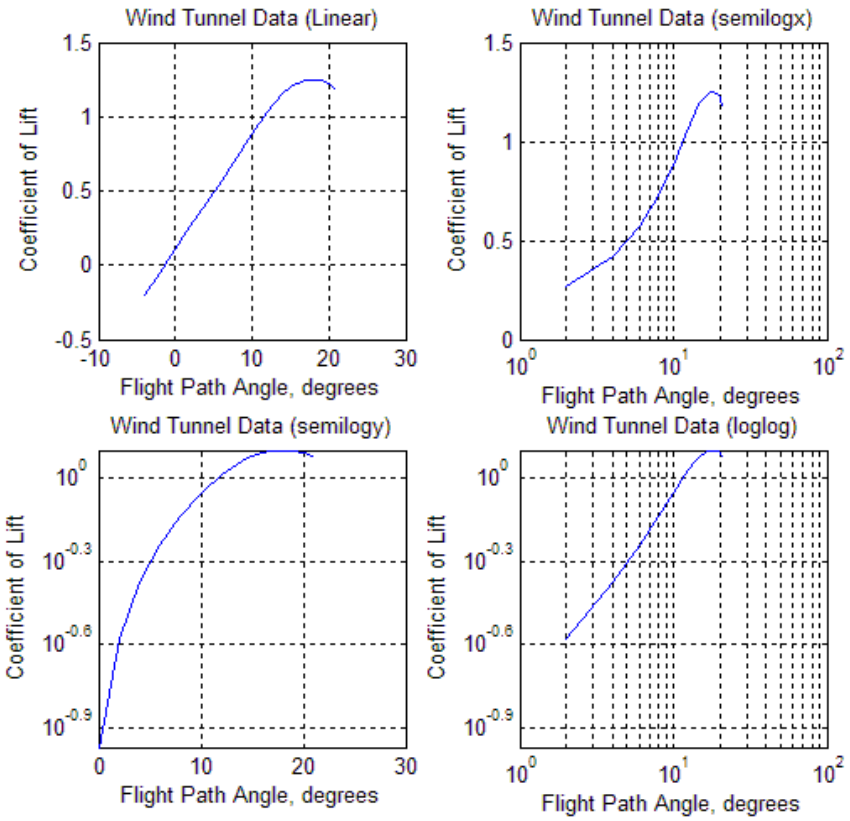
მნიშვნელოვანია გავიხსენოთ, რომ ნულის ტოლი ან ნულზე ნაკლები სიდიდის ლოგარითმი არ არსებობს, ამიტომ თუ მონაცემები, რომლებიც უნდა ავაგოთ **semilog** ან **loglog** ბრძანებით, შეიცავს უარყოფით ან ნულის ტოლ მნიშვნელობებს, MATLAB მიგვითითებს შეცდომაზე და გვამცნობს, რომ ეს მნიშვნელობები გამოტოვებული იქნება გრაფიკის აგების დროს.

თუ x ან y გრაფიკულ ბრძანებებში მატრიცაა, მაშინ გრაფიკი აიგება მატრიცის თითოეული სტრიქონის ან სვეტის მიმართ ერთიდაიგივე გრაფიკზე.

თუ x და y ერთნაირი ზომის მატრიცებია, აიგება x მატრიცის ყოველი სვეტი y შესაბამისი სვეტის მიმართ.

თითოეულ ამ ბრძანებათაგანს შეიძლება ჰქონდეს ერთი არგუმენტი. ამ შემთხვევაში x ვექტორის მნიშვნელობები აიგება ისეთი y ვექტორის მიმართ, რომლის ელემენტებია x მნიშვნელობათა შესაბამისი ინდექსი.

გავიხსენოთ, რომ მეორე თავში ავაგეთ თვითმფრინავის ფრენის ტრაექტორიის მნიშვნელობები lift (ამწევი ძალის) კოეფიციენტის მიმართ. ნახ 7.3 –ზე წარმოდგენილია ამ მნიშვნელობათა გრაფიკები წრფივი, semilogx, semilogy და loglog ბრძანებებით:



ნახ 7.3. წრფივი და ლოგარითმული გრაფიკები

პირველი მათგანი წარმოადგენს წრფივ გრაფიკს:

```
plot(x,y),...
title('Wind Tunnel Data (Linear)'),...
xlabel('Flight Path Angle, degrees'),...
ylabel('Coefficient of Lift'),...
grid
```

გავიხსენოთ, რომ x ვექტორის რამდენიმე მნიშვნელობა უარყოფითია, ამიტომ MATLAB ბრძანებათა ფანჯარაში ლოგარითმული გრადაციის შემთხვევაში მივიღებთ შესაბამის შეტყობინებას:

```
Warning: Negative data ignored.
```

იმისათვის, რომ უფრო თვალსაჩინო გრაფიკი შევარჩიოთ სასურველია ავაგოთ ჩვენი მონაცები სხვადასხვა ბრძანებათა გამოყენებით.

სავარჯიშო

შეადგინეთ 100 ელემენტიანი x ვექტორი 0 დან ბიჯით 0,5 და გამოითვალეთ შესაბამისი y როგორც x –ის ფუნქცია:

$$y = 5x^2$$

1. ააგეთ ამ მონაცემთა წრფივი გრაფიკი
2. ააგეთ ამ მონაცემთა გრაფიკი ლოგარითმული გრადაციით x-ის მიმართ
3. ააგეთ ამ მონაცემთა გრაფიკი ლოგარითმული გრადაციით y-ის მიმართ
4. ააგეთ ამ მონაცემთა loglog გრაფიკი
5. შეადარეთ გრაფიკები ერთმანეთს აღწერეთ თითოეულის უპირატესობა და ნაკლი

4.2 პოლარული გრაფიკი

ზოგჯერ მოცემული გვაქვს პოლარული კოორდინატები – სიდიდე და შესაბამისი კუთხე, მაგალითად ვზომავთ სინათლის ინტენსივობას წყაროს ირგვლივ. შეგვიძლია ინფორმაცია წარმოვადგინოთ როგორც რაიმე ფიქსირებული ღერძის მიმართ ათვლილი კუთხე და ინტენსივობა ამ მიმართულებით. ასეთი მონაცემები უფრო თვალსაჩინო სურათს მოგვცემს თუ ავაგებთ პოლარულ გრაფიკს. პოლარული გრაფიკი ასევე თვალსაჩინოდ წარმოგვიდგენს კომპლექსურ რიცხვებს.

4.2.1 პოლარული კოორდინატები

წერტილი კოორდინატთა პოლარულ სისტემაში განსაზღვრულია ორი სიდიდით – კუთხე θ და მოდული. კუთხის მნიშვნელობები ხშირად მოცემულია როგორც სიდიდეები 0 და 2π რადიანს შორის ($0 - 360^0$). მოდული დადებითი რიცხვია და წარმოადგენს მანძილს კოორდინატთა სათავიდან ამ წერტილამდე მოცემული მიმართულებით.

4.2.2 ბრძანება **polar**

MATLAB –ში პოლარული გრაფიკი აიგება ბრძანებით **polar**. მისი არგუმენტებია კუთხე (θ) და მოდული (r), მანძილი.

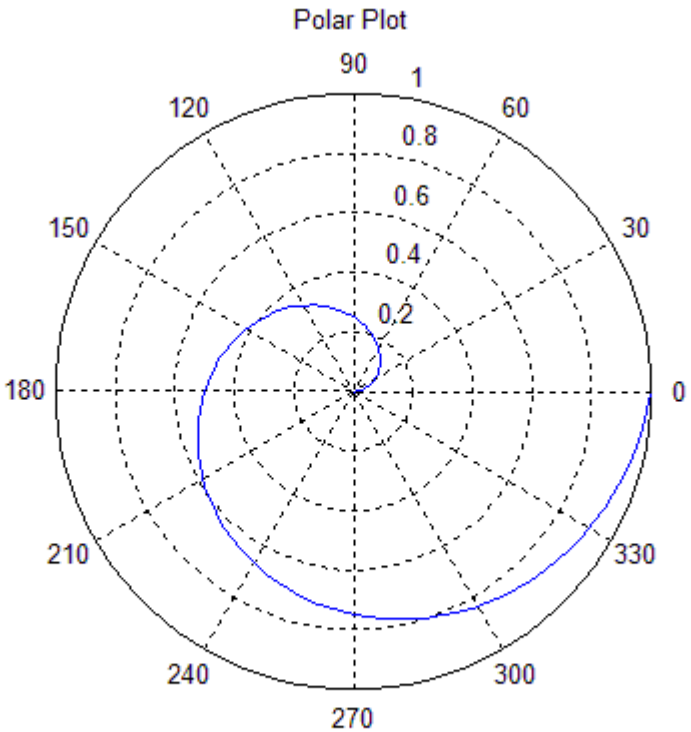
polar(theta,r)	ეს ბრძანება ააგებს პოლარულ გრაფიკს θ კუთხისა და შესაბამისი მოდულისათვის-r

თუ რომელიმე არგუმენტთაგანი მატრიცაა, მაშინ ვექტორი აიგება მატრიცის სვეტების ან სტრიქონების მიმართ, ერთდროულად მივიღებთ იმდენ მრუდს, რამდენი სტრიქონიც (სვეტიც) შედის მატრიცაში.

თუ ორივე არგუმენტი ერთნაირი ზომის მატრიცაა, ერთი მათგანის სვეტები აიგება მეორის შესაბამისი სვეტების მიმართ.

დავუშვათ გვინდა ავაგოთ მრუდი წრეწირის წერტილებისა ზრდადი რადიუსით. შევქმნით ვექტორს θ, რომლის ელემენტებია რადიანებში გამოსახული კუთხე 0 დან 2π-მდე. შევქმნათ ვექტორი r, რომლის ელემენტები იქნებს რადიუსების შესაბამისი მნიშვნელობები 0 დან 1-მდე. ნახ 7.4 შეიცავს გრაფიკს, რომელიც აიგება ბრძანებათა შემდეგი მწკრივით:

```
theta = 0:2*pi/100:2*pi;
r = theta/(2*pi);
polar(theta,r),...
    title('Polar Plot'),...
    grid
```



ნახ 7.4. პოლარული გრაფიკი წრე ზრდადი რადიუსით

4.3 მართკუთხა/პოლარული გარდაქმნა

ხშირად საჭიროა მონაცემები კოორდინატთა ერთი სისტემდან მეორეში გადავიყვანოთ. ტრიგონომეტრიის გამოყენებით ასეთი ტრანსფორმაცია გამოისახება ფორმულებით:

პოლარული კოორდინატების გადაყვანა მართკუთხა სისემაში:

$$x = r\sin\theta$$
$$y = r\cos\theta$$

მართკუთხა კოორდინატების გადაყვანა პოლარულ სისტემაში:

$$r = \sqrt{x^2 + y^2}$$

$$\theta = \tan^{-1}\left(\frac{y}{x}\right)$$

ტანგენსის შებრუნებული სიდიდის გამოთვლისას სიფრთხილეა საჭირო, რომ სწორად შევარჩიოთ კუთხის მნიშვნელობა. თუ MATLAB-ს იყენებთ ამ გარდაქმნის განსახორციელებლად, შეგიძლიათ თავი დაიზღვიოთ **atan2** ფუნქციის გამოყენებით.

სავარჯიშო

გდაიყვანეთ მოცემული წერტილები კოორდინატთა მართკუთხადან პოლარულ სისტემაში:

1. (3, -2)
2. (0, -1)
3. (-2,0)
4. (0.5, 1)

გადაიყვანეთ შემდეგი წერტილები პოლარულიდან მართკუთხა სისტემაში:

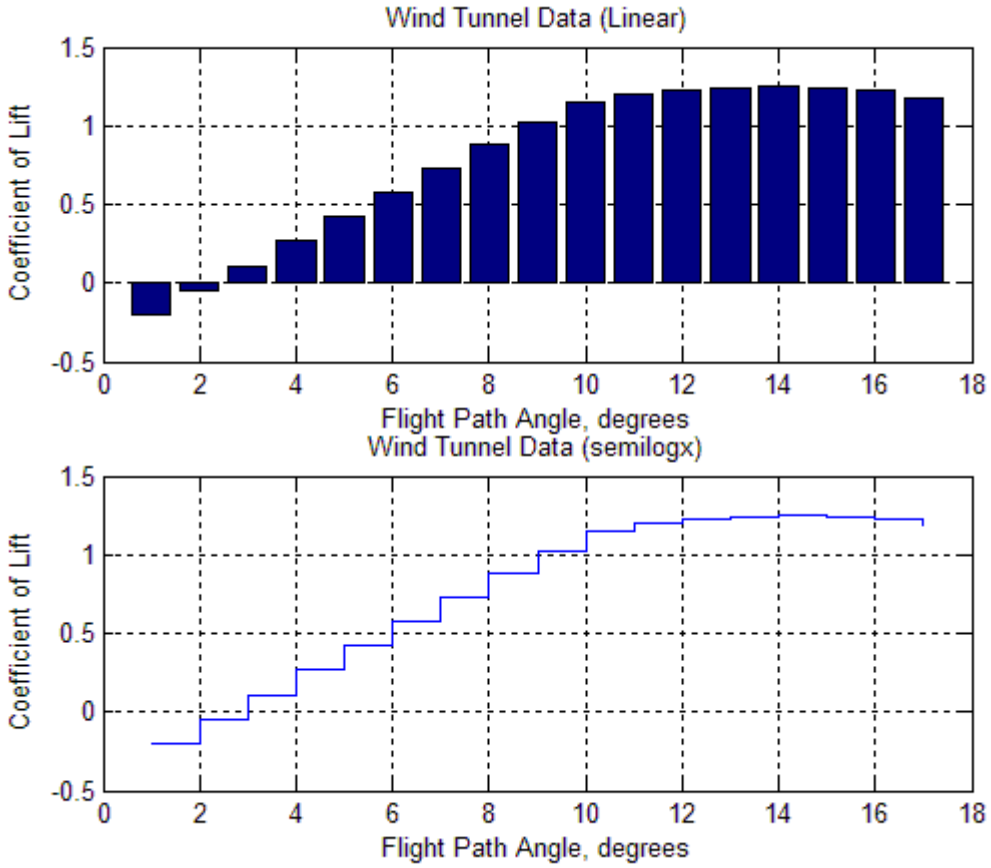
5. (π, 1)
6. (π/2, 0)
7. (2.3, 0.5)
8. (0.5, 0.5)

4.4 bar და stairs გრაფიკები

ნახ 7.5 წარმოადგენს **bar** და **stairs** გრაფიკებს, აგებულს აეროდინამიკური გვირაბის ექსპერიმენტის მონაცემებისათვის, რომელიც უკვე რამდენიმეჯერ შეგვხვდა განხილულ მაგალითებში.

bar(y)	ააგებს bar გრაფიკს y მნიშვნელობათათვის
bar(x,y)	ააგებს bar გრაფიკს y ვექტორის ელემენტებისათვის იმ მნიშვნელობებზე, რომელიც მოცემულია x ვექტორის სახით
stairs(y)	ააგებს stair გრაფიკს y ვექტორის მნიშვნელობათათვის.
stairs(x,y)	ააგებს stair გრაფიკს y ვექტორის ელემენტებისათვის იმ მნიშვნელობებზე, რომელიც მოცემულია x ვექტორის სახით

```
subplot(2,1,1), bar(y)
    title('Wind Tunnel Data (Linear)'),...
    xlabel('Flight Path Angle, degrees'),...
    ylabel('Coefficient of Lift'),...
    grid
    subplot(2,1,2),stairs(y)
    title('Wind Tunnel Data (semilogx)'),...
    xlabel('Flight Path Angle, degrees'),...
    ylabel('Coefficient of Lift'),...
    grid
```



ნახ 7.5. bar და stairs გრაფიკები

4.5 გრაფიკული ოფციები

MATLAB შეიცავს სხვადასხვა ოფციებს გრაფიკის გასაფორმებლად, გასაუმჯობესებლად. განვიხილოთ ზოგიერთი მათთგანი, რომელიც შემდგონში ხშირად გამოგვადგება.

4.5.1 რამდენიმე მრუდის აგება ერთიდაიგივე ნახაზზე

არსებობს ერთიდაიგივე ნახაზზე რამდენიმე მრუდის აგების სამი გზა. ერთ-ერთი მათგანია: **plot** ბრძანების არგუმენტად ავიღოთ მატრიცები.

მეორე გზაა plot ბრძანებას მივცეთ რამდენიმე არგუმენტი, მაგალითად:

plot(x,y,w,z)

სადაც x, y, w, z ვექტორებია. როცა ამ ბრძანებას მივცემთ, MATLAB ააგებს ორივე გრაფიკს ერთ ნახაზზე (ერთ გრაფიკულ ფანჯარაში). ამ მეთოდის უპირატესობა ისაა, რომ საჭირო არაა წერტილების რაოდენობა ერთმანეთს ემთხვეოდეს. MATLAB ავტომატურად შეარჩევს მათთვის წირის განსხვავებულ ფორმას და ფერს.

მესამე გზა ერთიდაიგივე ნახაზზე სხვადასხვა მონაცემთა აგებისა ხორციელდება ბრძანებით **hold.** ეს ბრძანება ინარჩუნებს გრაფიკულ ფანჯარას აქტიურ მდგომარეობაში და ყოველი შემდგომი გრაფიკული ბრძანება ახალ გრაფიკს უმატებს მას, **hold** ბრძანების ხელმეორედ მიცემა შეწყვეტს ამ პროცესს.

4.5.2 წირის და აღნიშვნის სტილი

ბრძანება **plot(x,y)** აწარმოებს წირის აგებას, რომელიც აერთებს x და y ვექტორებით მოცემულ წერტილებს. შესაძლებელია აირჩიოთ და მიუთითოთ წირის ფორმა (წყვეტილი, წერტილოვანი და წყვეტილწერტილოვანი) და ფერი. შეგიძლიათ წირით სულაც არ შეაერთოთ წერტილები. ასეთ შემთხვევაში მხოლოდ ვექტორებით განსაზღვრული წერტილები აღინიშნება ნახაზზე. შეგვიძლია მივუთითოთ აღნიშვნა წერტილის ადგილზე: ვარსკვლავი, წრე, სამკუთხედი და სხვა. შემდეგი ცხრილი შეიცავს წირისა და წერტილის აღნიშვნების მრავალფეროვან არჩევანს:

წირის ტიპი	განმსაზღვრელი	წერტილის ტიპი	განმსაზღვრელი
უწყვეტი	-	წერტილი	.
წყვეტილი	--	პლუსი	+
წერტილოვანი	:	ვარსკვლავი	*
წყვეტილ-წერტილოვანი	-.	წრე	O
		x - აღნიშვნა	x

საილუსტრაციოდ მოგვყავს მაგალითი. ბრძანება უწყვეტი წირით აერთებს წერტილებს, რომელთა კოორდინატები წარმოდგენილი x და y ვექტორების სახით და შემდეგ დასვამს ამ წირზე შესაბამის წერტილებს აღნიშვნით x:

```
plot(x, y, x, y, 'x')
```

გარდა ამისა, შეგიძლიათ შეარჩიოთ და მიუთითოთ წირის ფერი შემდეგი ცხრილის მიხედვით:

წირის ტიპი	განმსაზღვრელი
წითელი	**r**
მწვანე	**g**
ლურჯი	**b**
თეთრი	**w**
უხილავი	**i**

შემდეგი ბრძანება უწყვეტი ლურჯი წირით აერთებს წერტილებს, რომელთა კოორდინატები წარმოდგენილი x და y ვექტორების სახით და შემდეგ დასვამს ამ წირზე შესაბამის წერტილებს წითელი ფერის x ნიშნით:

```
plot(x, y, 'b', x, y, 'xr')
```

4.5.3 გრადაცია

MATLAB ავტომატურად ირჩევს ღერძების გრადაციას. მაგრამ შესაძლებელია მივუთითოთ ღერძების საზღვრები ბრძანებით **axis:**

axis	ეს ბრძანება გრაფიკულ ფანჯარაზე 'გაყინავს' ღერძების არსებულ გრადაციას, ავტომატურ რეჟიმში დასაბრუნებლად იგივე ბრძანებას ვიმეორებთ
axis(v)	v 4 ელემენტიანი ვექტორია, რომელიც შეიცავს გრადაციის სიდიდეებს [xmin, xmax, ymin, ymax]
axis('square')	განსაზღვრავს ღერძების თანაფარდობას (კვადრატული)
axis('normal')	განსაზღვრავს ღერძების თანაფარდობას (ნორმალური)

ეს ბრძანებები განსაკუთრებით სასარგებლოა სხვადასხვა გრადაციით(scaling) აგებული ნახაზების შესადარებლად.

4.5.4 ბრძანება subplot

ბრძანება **subplot** საშუალებას გვაძლევს გრაფიკული ფანჯარა დავყოთ რამდენიმე ქვეფანჯარად. ნახ 7.3 და ნახ 7.5 შექმნილია ამ ბრძანების გამოყენებით. **subplot** გააჩნია 3 არგუმენტი subplot (m n p), m,n განსაზღვრავს რამდენ ქვეფანჯრადაა დაყოფილი გრაფიკული ფაბჯარა და რა რიგით, ხოლო p მიუთითებს რომელ ფანჯარაში აიგოს მოცემული გრაფიკი. ფანჯრები დანომრილია მარცხნიდან მარჯვნივ, ზემოდან ქვემოთ. მაგალითად შემდეგი ბრძანებები განსაზღვრავს, რომ გრაფიკული ფანჯარა დაყოფილია ორ ქვეფანჯარად – ზედა და ქვედა, ხოლო მოცემული გრაფიკი აიგება ზედა ფანჯარაში:

```
subplot(211), plot(x,y)
```

4.6 ეკრანის კონტროლი

როგორც ვიცით MATLAB-ს ორი ძირითადი ფანჯარა აქვს ბრძანებათა ფანჯარა და გრაფიკული ფანჯარა შემდეგი ბრძანებები საშუალებას გვაძლევს შევარჩიოთ და გავასუფთაოთ ფანჯარა:

shg	ეკრანზე გამოიყვანს გრაფიკულ ფანჯარას
ნებისმიერი კლავიში	დააბრუნებს ეკრანზე ბრძანებათა ფამჯარას

clc	ასუფთავებს ბრძანებათა ფანჯარას
clf	ასუფთავებს გრაფიკულ ფანჯარას
home	ასუფთავებს ბრძანებათა ფანჯარის ხილულ ნაწილს ეკრანზე და კურსორი გადადის ბრძანებათა ფანჯარის ზედა მარცხენა კუთხეში

4.6.1 ბრძანება ginput

ეს ბრძანება საშუალებას გვაძლევს ავიღოთ კოორდინატები პირდაპირ გრაფიკული ფანჯარიდან მაუზის ან ისრიანი კლავიშების საშუალებით:

[x,y]=ginput	საშუალებას გვაძლევს შევარჩიოთ წერტილთა შეუზღუდავი რაოდენობა გრაფიკული ფანჯარიდან მაუზის ან კლავიშების საშუალებით. შეიქმნება x, y ვექტორები კოორდინატთა შესაბამისი მნიშვნელობებით. ბრძანების მოქმედებას წყვეტს კლავიში return key
[x,y]=ginput(n)	საშუალებას გვაძლევს შევარჩიოთ n წერტილი გრაფიკული ფანჯარიდან მაუზის ან კლავიშების საშუალებით. შეიქმნება x, y ვექტორები კოორდინატთა შესაბამისი მნიშვნელობებით. ბრძანების მოქმედებას წყვეტს კლავიში return key

4.6.2 გრაფიკის დაბეჭდვა

print	დაბეჭდავს მაღალი გარჩევის გრაფიკს პრინტერზე, ან შეინახავს მას როგორც .pdf ფაილს დისკზე.

პრობლემა - მეტეოროლოგიური რაკეტის ტრაექტორია

მეტეოროლოგიური რაკეტა გამოიყენება ატმოსფეროს სხვადასხვა ფენებში მიმდინარე პროცესების შესახებ მონაცემთა შესაგროვებლად. მაგალითად ოზონის შემცველობის განსაზღვრა. რაკეტაზე სხვა ხელსაწყოებთან ერთად დამაგრებულია ტელემეტრული სისტემა, როლის საშუალებითაც ხდება მონაცემების გადმოგზავნა დედამიწაზე. ყოველ ანათვლს თან ახლავს მონაცემები თავად რაკეტის შესახებ ამ მომენტისათვის: სიმაღლე, სიჩქარე და აჩქარება.

დავუშვათ გვაქვს მონაცემთა ფაილი, რომელიც შეიცავს ინფორმაციას იონოსფეროს გამოსაკვლევად გაშვებული ორსაფეხურიანი (two-stage) მეტეოროლოგიური რაკეტის სიმაღლის, სიჩქარის და აჩქარების შესახებ. ვიცით, რომ რაკეტის პირველი საფეხურის საწვავი დაიწვა 35 წამში, რის შემდეგაც რაკეტის სიჩქარემ მიაღწია 1 250 მეტრს წამში. ამის შემდეგ 2 წუთის განმავლობაში რაკეტა განიცდის თავისუფალ ვარდნას და აღწევს იონოსფეროს დაბალ ფენებს 100 კილომეტრის სიმაღლეზე. ამ დროისათვის გრავიტაცია შეანელებს რაკეტის სიჩქარეს 100 მეტრ/წამამდე. ამის შემდეგ ირთვება მეორე საფეხურის საწვავის მექანიზმი, რომელიც გამოიწვევს რაკეტის აჩქარებას. იგი გაიჭრება იონოსფეროს მაღალ ფენებში. უნდა ავაგოთ მრუდი ფაილში ჩაწერილი მონაცემების მიხედვით, რათა შევადაროთ ისინი თერიულად გათვლილ ტრაექტორიას.

1. **ამოცანის დასმა**

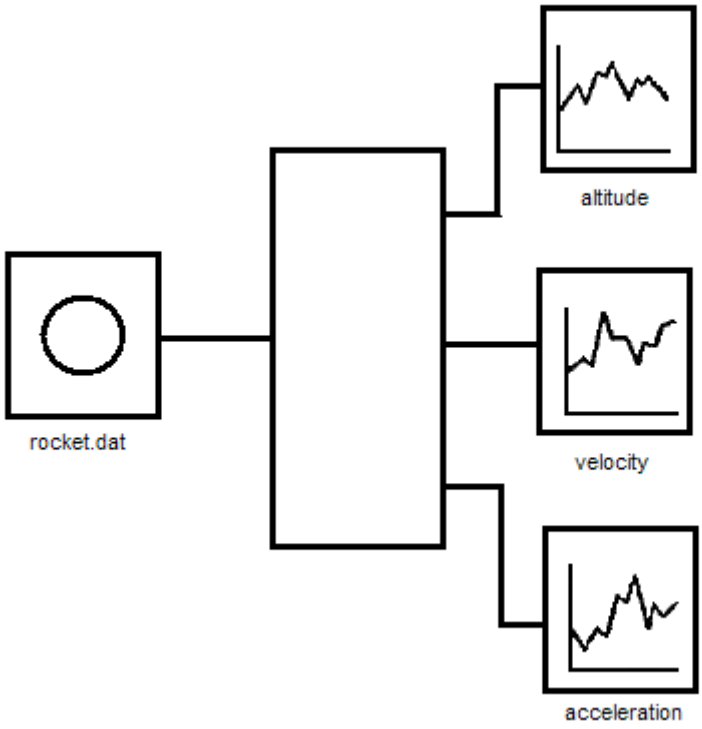
ავაგოთ რაკეტის სიმაღლის, სიჩქარის, და აჩქარების მრუდი, ფაილიდან წაკითხულ მონაცემთა მიხედვით.

2. INPUT/OUTPUT აღწერა

ცხრილი შეიცავს მონაცემებს რაკეტის სიმაღლის, სიჩქარის და აჩქარების მნიშვნელობებს დროის შესაბამისი მომენტუსათვის.

დრო [წმ]	სიმაღლე [მ*10^5]	სიჩქარე [მ/წმ]	აჩქარება[მ/$წმ^2$]
10	0	500	42
20	0.1100	895	40
30	0.2500	1245	10
40	0.3400	1250	-9
50	0.4200	1147	-10
60	0.5300	1046	-10
70	0.6200	920	-10
80	0.6900	824	-10
90	0.7500	711	-10
100	0.8300	609	-10
110	0.9200	501	-10
120	0.9900	400	-10
130	1.0300	286	-10
140	1.0700	190	-10
150	1.1300	90	-5
160	1.1500	264	30
170	1.2100	741	56
180	1.3000	1381	78
190	1.3700	2141	94
200	1.6100	3195	118
210	2.0000	3791	19
220	2.3500	3811	-10
230	2.7600	3698	-10
240	3.3000	3560	-10

ნახ 7.6 ნახაზზე წარმოდგენილია განსახილველი ამოცანის INPUT/OUTPUT დიაგრამა, რომელიც გვიჩვენებს, რომ საწყისი მონაცემები ფაილის სახითაა მოცემული, ხოლო შედაგად უნდა მივიღოთ მონაცემთა მიხედვით აგებული გრაფიკები.



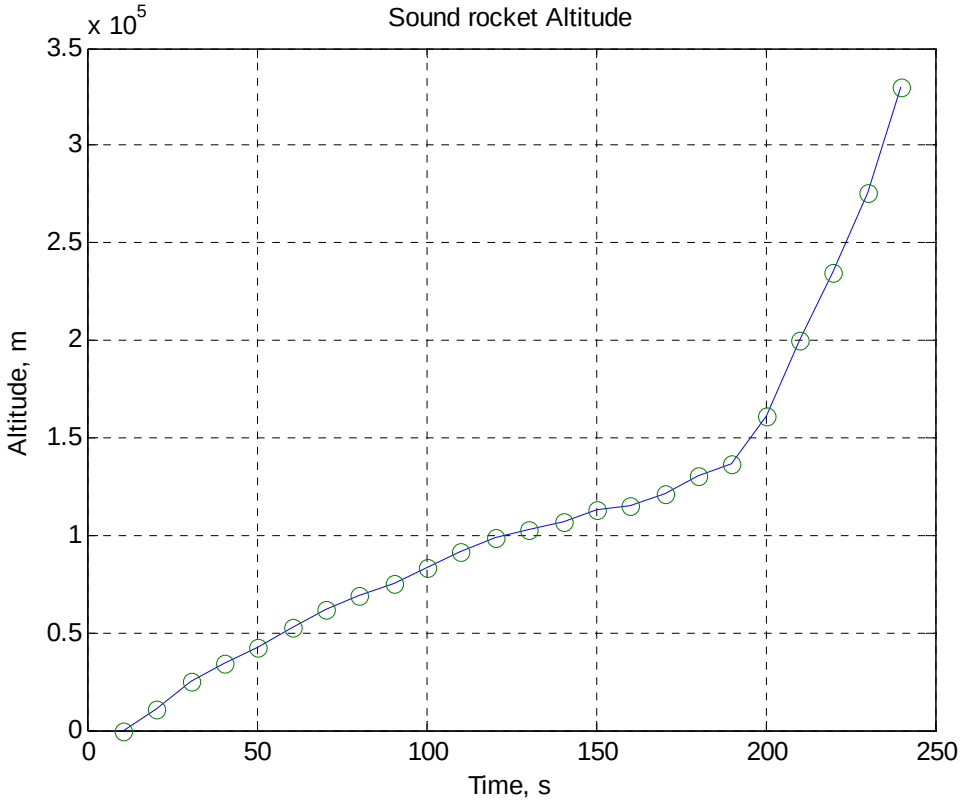
ნახ 7.6. I/O დიაგრამა

3. სახელდახელო ამოხსნა

რადგან ამ ამოცანაში არაფერს არ ვითვლით, გამოსათვლელიც არაფერია. მაგრამ შეგვიძლია ამოვიღოთ ფაილიდან რამდენიმე მონაცემი და მიღებული შედეგის მიხედვით გავაკეთოთ დასკვნა.

4. MATLAB ამოხსნა

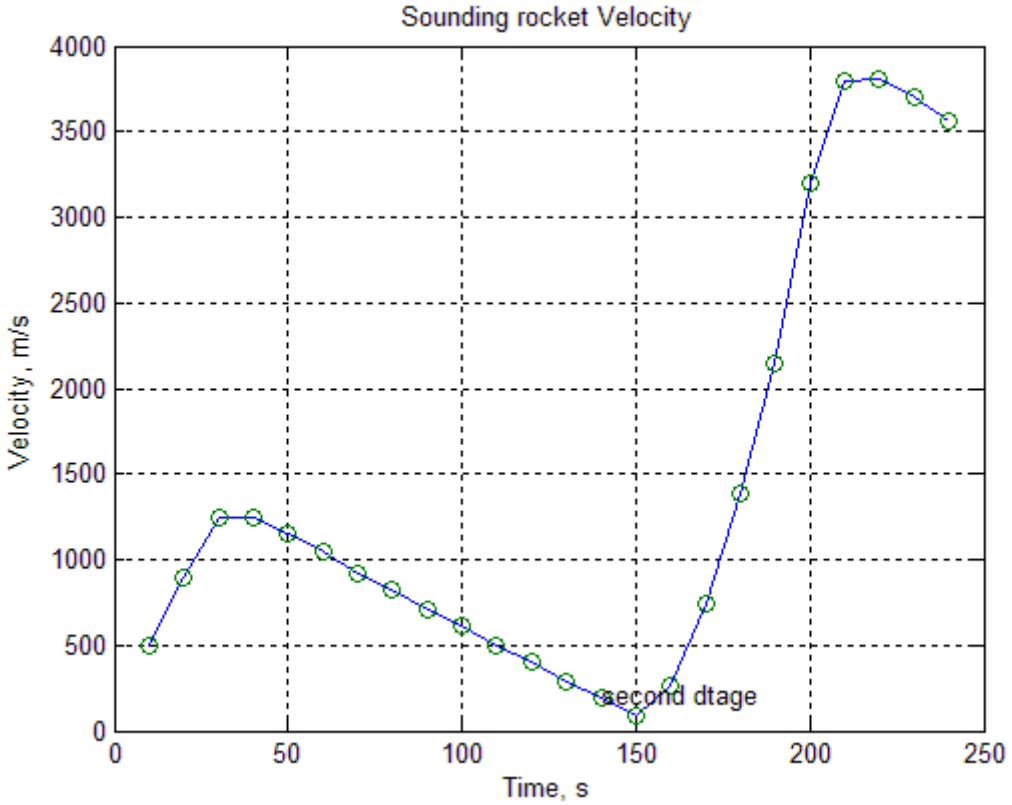
MATLAB მრავალფეროვანი გრაფიკული შესაძლებლობების გამო ასეთი გრაფიკების აგება მარტივად ხდება. ავაგოთ წირი და შემდეგ დავსვათ წერტილთა შესაბამისი აღნიშვნები.



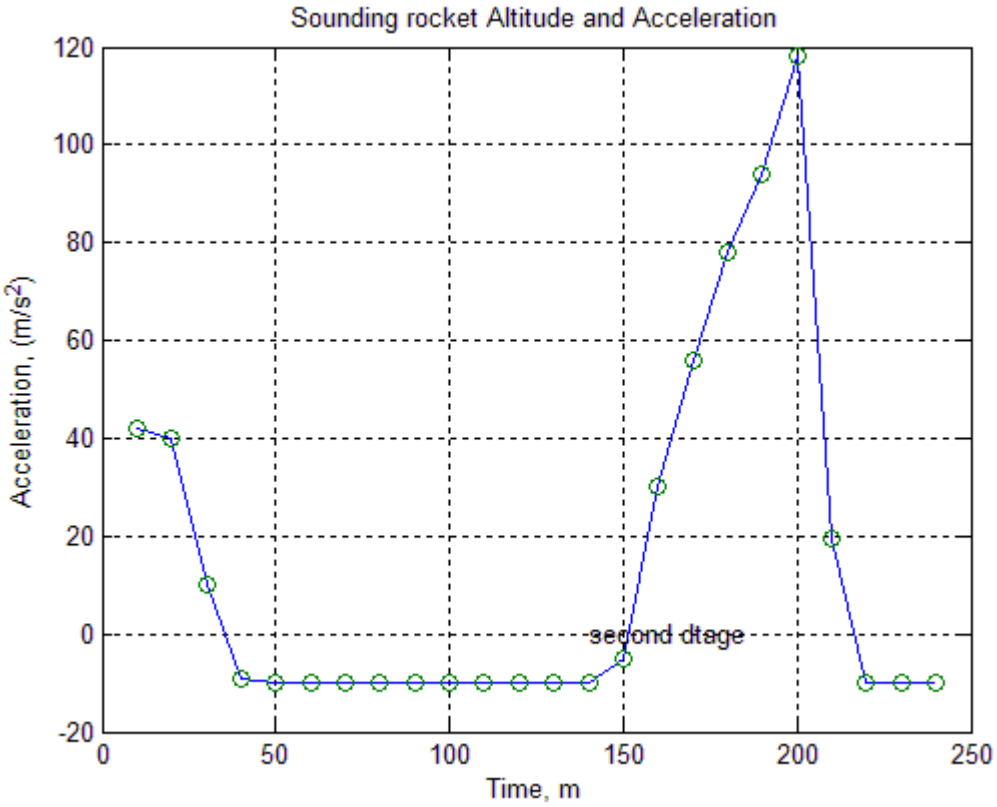
ნახ 7.7. მეტეოროლოგიური რაკეტის სიმაღლე

```
%
%     This program generates plots of the altitude,
%     velocity, and acceleration of sounding rocket
%
clear; clc;
load rocket.dat
time=rocket(:,1);
alt=rocket(:,2);
vel=rocket(:,3);
acc=rocket(:,4);
%
%     These commands generate and label
%     a plot of the altitude data
%

plot(time,alt,time,alt,'o'),...
title('Sound rocket Altitude'),...
xlabel('Time, s'),...
ylabel('Altitude, m'),...
grid
pause
%
%     These commands generate and label
%     a plot of the velocity data
```

```
%
plot(time,vel,time,vel,'o'),...
title('Sounding rocket Velocity'),...
xlabel('Time, s'),...
ylabel('Velocity, m/s')
text(140,200,'second dtage')
grid
pause
%
%    These commands generate and label
%    a plot of the acceleration data
%
plot (time,acc,time,acc,'o')
title('Sounding rocket Altitude and Acceleration'),...
xlabel('Time, m'),...
ylabel('Acceleration, (m/s^2)')
text(140,0,'second dtage')
grid
```



ნახ 7.8. მეტეოროლოგიური რაკეტის სიჩქარე



ნახ 7.9. მეტეოროლოგიური რაკეტის აჩქარება

5. შემოწმება

მონაცემები ჩაწერილია ფაილში **rocket.dat**. 7.7 – 7.9 ნახაზზე წარმოდგენილია შედეგად მიღებული გრაფიკები მეტეოროლოგიური რაკეტის სიმაღლის, სიჩქარისა და აჩქარებისათვის შესაბამისად. სიჩქარის გრაფიკიდან კარგად ჩანს, რომ თავდაპირველად იგი იზრდება, პირველი საფეხურის საწვავის დაწვის შემდეგ – თანდათან კლებულობს, ხოლო როცა მეორე საფეხურის საწვავი იწყებს წვას, ისევ მომატებას იწყებს. აჩქარების გრაფიკიდან ადვილად შეამჩნევთ აჩქარებას, რომელიც გამოწვეულია პირველი საფეხურის და მეორე საფეხურის გამო. საწვავის ამოწურვის სტადიაში ორივე შემთხვევაში რაკეტის აჩქარება –9.5 მ/წმ-ია, რაც გრავიტაციით გამოწვეული თავისუფალი ვარდნის აჩქარებაა. ნახ. 7.8 და 7.9 **text** ბრძანების საშუალებით მივუთითეთ მეორე საფეხურის დასაწყისის შესაბამისი წერტილი გრაფიკზე.

4.7 სამგანზომილებიანი გრაფიკი

MATLAB –ში სამგანზომილებიანი გრაფიკის სხვადასხვა გზა არსებობს. შესაძლებელია ავაგოთ ბადურა (mesh) ზედაპირი სამგანზომილებიან სივრცეში. შეგვიძლია ასეთ ზედაპირს შევხედოთ გრაფიკულად ნებისმიერი მიმართულებიდან, შეგვიძლია განვსაზღვროთ ღერძების გრადაცია. სამგანზომილებიანი ზედაპირი შეიძლება ავაგოთ როგორც ზედაპირის კონტური სხვადასხვა ჭრილში. სამგანზონმილებიანი გრაფიკის ყველა ფორმა მნიშვნელოვან ინფორმაციას იძლევა სამგანზომილებიან მონაცემთა ანალიზისათვის.

4.7.1 ბადურა ზედაპირი (mesh ზედაპირი)

mesh ზედაპირი აიგება მატრიცის სახით წარმოდგენილი მონაცემების საფუძველზე. მატრიცის თითოეული ელემენტი შესაბამება წერტილს ბადეზე.

იმისათვის, რომ შევქმნათ მონაცემები სამგანზომილებიანი ზედაპირისათვის, პირველ რიგში განვსაზღვრავთ დამოუკიდებელი ცვლადების x და y სიდიდეების მწკრივს, შემდეგ გამოვითვლით z სიდიდეებს, რომელიც x და y –ს ფუნქციაა და წარმოადგენს სამგანზომილებიან ზედაპირს. x და y ისე ავირჩევთ, რომ მათი მნიშვნელობები თანაბრად იყოს განაწილებული x-y სიბრტყეზე. მაგალითად, დავუშვათ გვინდა ავაგოთ ფუნქცია

$$f(x,y) = z = \left|\sqrt{1 - x^2 - y^2}\right|$$

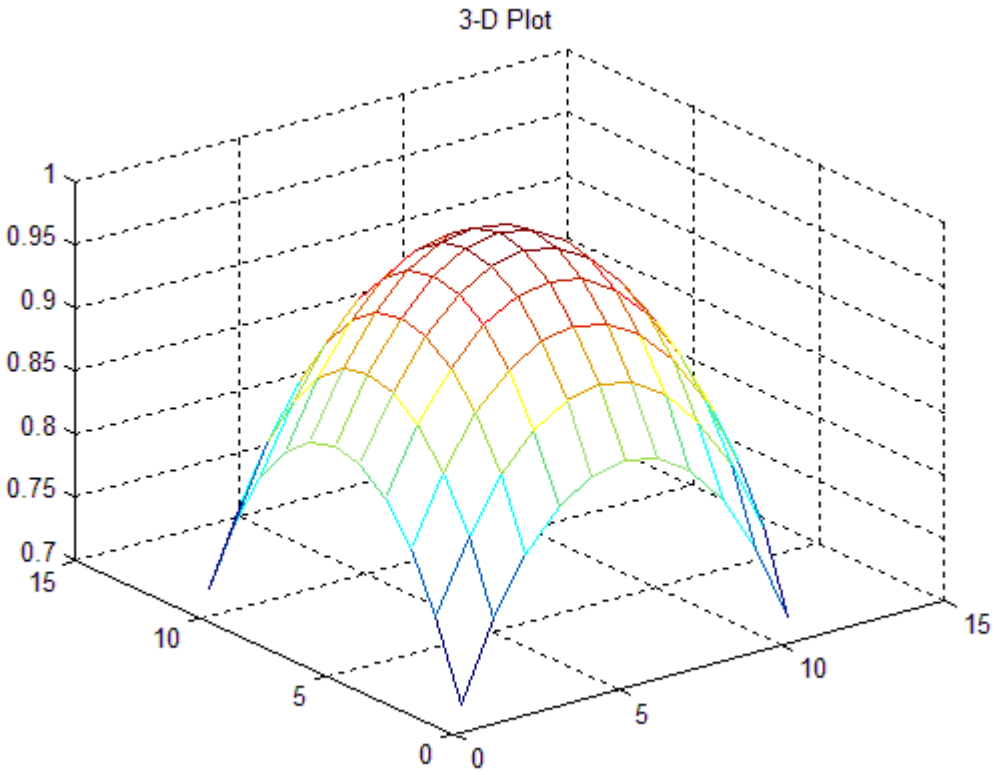
ისე, რომ $-0.5 \le x \le 0.5$ და $-0.5 \le y \le 0.5$

ეს ფუნქცია წარმოადგენს ერთეულოვანი რადიუსის მქონე სფეროს ზედაპირის განტოლების სახეცვლილებას:

$$x^2 + y^2 + z^2 = 1$$

რადგან *f(x,y)* იყენებს კვადრატული ფესვის მხოლოდ დადებით მნიშვნელობებს, იგი ამ ფუნქციის მხოლოდ ზედა ნახევარს წარმოადგენს. იმისათვის, რომ ავაგოთ სამგანზომილებიანი ზედაპირი შემდეგნაირად უნდა მოვიქცეთ:

```
for m =  1:11
    x = (m-6)*0.1;
    for n = 1:11
        y = (n-6)*0.1;
        z(m,n) = sqrt(abs(1 - x.^2 - y.^2));
    end
end
mesh(z),...
title('3-D Plot')
grid
```



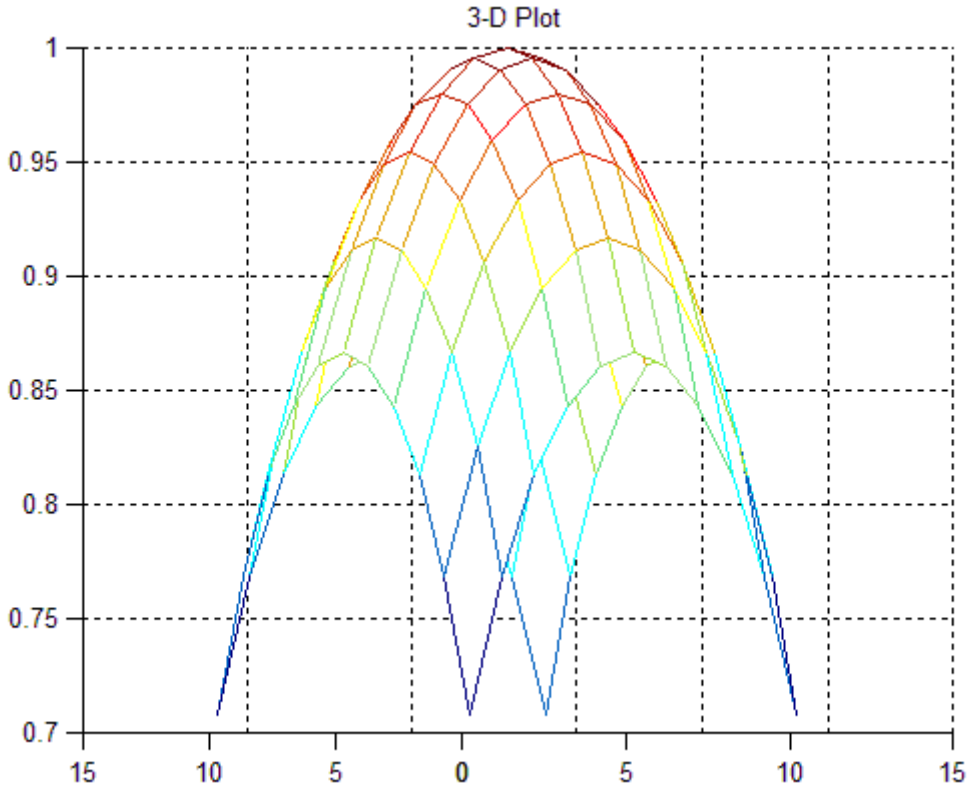
ნახ 7.10. სფეროს ნაწილის სამგანზომილებიანი გრაფიკი

z ვექტორის აგების მეორე გზაც არსებობს. ვსარგებლობთ ბრძანებით meshgrid, რომლის არგუმენტები x და y ვექტორის მნიშვნელობებია. იგი აწარმოებს 2 მასივს x და y მნიშვნელობების მიხედვით. სფეროს ზემოთაღწერილი ნაწილი ასეც შეგვიძლია ავაგოთ:

```
[X,Y]= meshgrid(-0.5:0.1:0.5,-0.5:0.1:0.5);
Z=sqrt(abs(1-X.^2-Y.^2));
mesh(Z)
grid
```

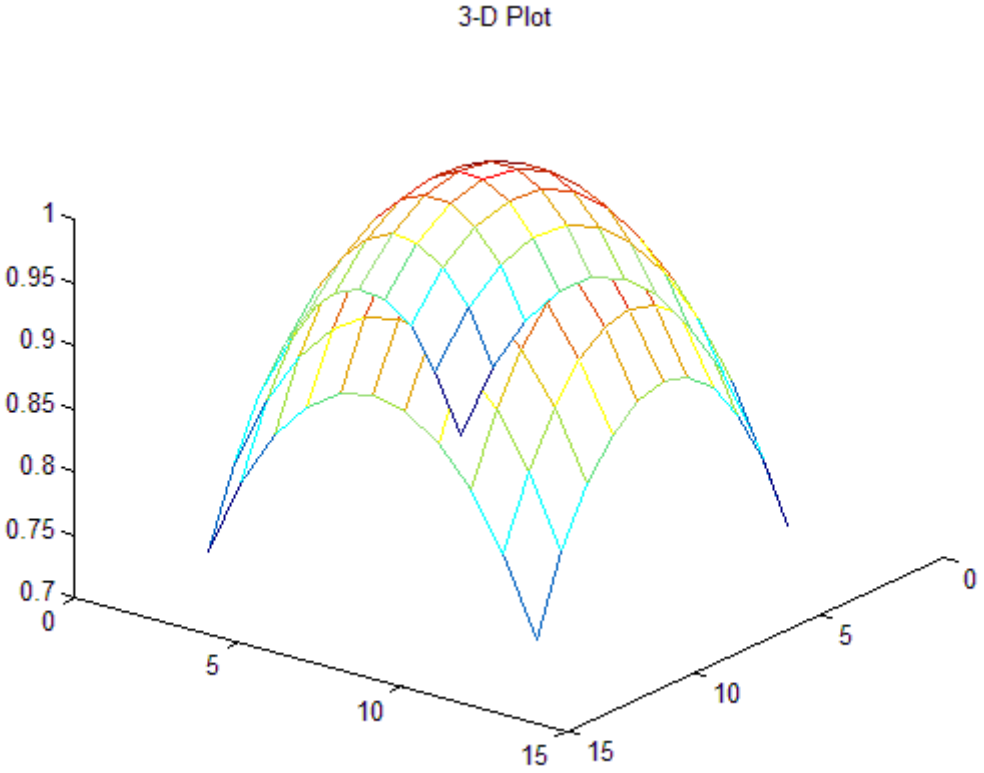
შეიძლება სამგანზომილებიანი გრაფიკის აგებისას საჭირო იყოს აგებულ ზედაპირს რაიმე გარკვეული მიმართულებიდან შევხედოთ. ხედვის მიმართულება განისაზღვრება გრადუსებში გამოხატული აზიმუტისა (ჰორიზონტალური) და სიმაღლის(ვერტიკალური) მიხედვით. აზიმუტი = 0, სიმაღლე = 0 შეესაბამება მატრიცის ქვედა მარჯვენა კუთხე. აზიმუტის დადებითი მიმართულება საათის ისრის საწინააღმდეგო მიმართულებაა, z ღეძის გასწვრივ. დადებითი სიმაღლეები გვიჩვენებს ზედაპირს ზემოდან, უარყოფითი – ქვემოდან.

თუ არ ვუთითბთ ხედვის კუთხეს მაშინ მას აქვს (default) მნიშვნელობა აზიმუტი –37.5 გრადუსი და სიმაღლე 30 გრადუსი ნახ 7.10. ხედვის კუთხე განისაზღვრება როგორც **mesh** ბრძანების მეორე არგუმენტი mesh(x, [–37.5, 0]). მაგალითად, ნახ 7.11 ხედვის კუთხეა [–37.5, 0]



ნახ 7.11. სფეროს ნაწილი. ხედვის კუთხე (-37.5, 0)

ნახ 7.12 ნახაზზე გრაფიკი ნაჩვენებია მიმართულებიდან: [-37.5, -30)



ნახ 7.12. სფეროს ნაწილი 30 გრადუსიანი სიმაღლიდან

სავარჯიშო

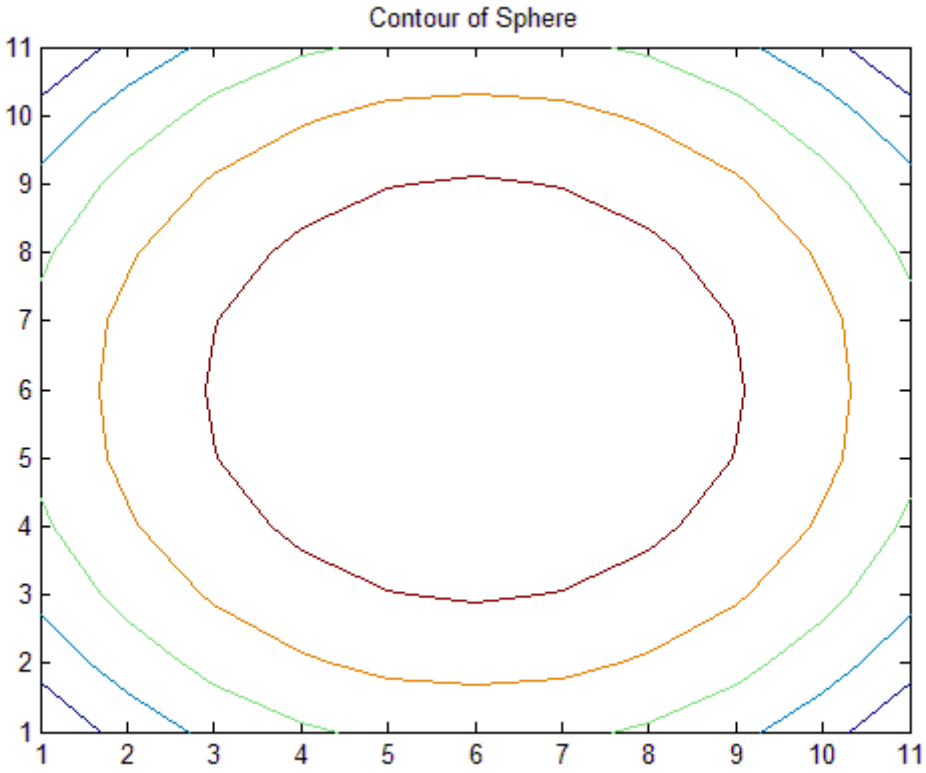
ააგე **mesh** ზედაპირი, რომელიც განისაზღვრება შემდეგი ფორმულით. ვიდრე ააგებდეთ. შეეცადეთ წარმოიდგინოთ, როგორი იქნება იგი.

1. $f(x,y) = \left|\sqrt{1-x^2-y^2}\right|$ *for* $0 \le x \le 0.5$ *and* *for* $0 \le y \le 0.5$
2. $f(x,y) = -\left|\sqrt{1-x^2-y^2}\right|$ *for* $-0.5 \le x \le 0.5$ *and* *for* $-0.5 \le y \le 0.5$
3. $f(x,y) = \left|\sqrt{1-x^2-y^2}\right|$ *for* $-0.5 \le x \le 0$ *and* *for* $0 \le y \le 0.5$
4. $f(x,y) = \left|\sqrt{1-x^2-y^2}\right|$ *for* $-1 \le x \le 1$ *and* *for* $-1 \le y \le 1$

4.7.2 კონტურული გრაფიკი

სიმაღლეების (elevation) რუქა შეიცავს წირების ჯგუფს, რომლებიც აერთებენ ერთნაირ სიმაღლეზე მყოფ წერტილებს. თუ დედამიწის ფიზიკურ რუქაზე ასეთი წირებით შევაერთებთ ზღვის დონიდან ერთი სიმაღლის მქონე წერტილებს, შეგვიძლია ვიმსჯელოთ მათი რეალური სიმაღლის შესახებ. ასეთი ტიპის რუქებს კონტურულს უწოდებენ. MATLAB-ის საშუალებით მატრიცის სახით მოცემული ზედაპირისათვის შეგვიძლია ავაგოთ მსგავსი კონტურული გამოსახულება. ამისათვის არსებობს ბრძანება **contour**:

contour(z)	აგებს z მატრიცით მოცემული ზედაპირის კონტურს. კონტურული წირების რაოდენობა და მათი მნიშვნელობები ავტომატრად შეირჩევა MATLAB –ის მიერ. ნახაზის ზედა მარცხენა კუთხე შეესაბამება სიდიდეს მდებარეობაში z(1,1)
contour(z,n)	აგებს z მატრიცით განსაზღვრული ზედაპირის n დონის კონტურს
contour(z,v)	აგებს z მატრიცით განსაზღვრული ზედაპირის კონტურულ გამოსახულებას კონტურული წირებით, რომელთა დონეები მოცემულია v ვექტორის სახით

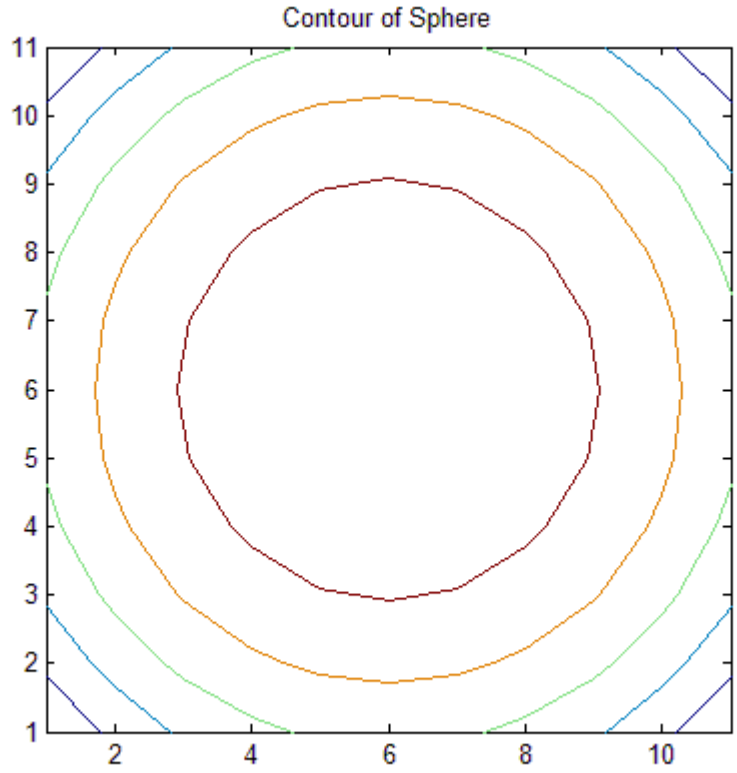


ნახ 7.13. გრაფიკი ზედაპირის 5 სხვადასხვა დონის კონტურით

ამ ბრძანებას შესაძლოა ჰქონდეს არგუმენტები, რომლებიც განსაზღვრავს კონტურთა დონეს და ღერძების მასშტაბირებას. შემდეგი ბრძანებები აგებს სფერული ზედაპირის 7.13 გრაფიკს:

```
[X,Y]= meshgrid(-0.5:0.1:0.5,-0.5:0.1:0.5);
Z=sqrt(abs(1-X.^2-Y.^2));
contour(Z,5),...
title('Contour of Sphere')
```

თუ ამ ბრძანებებს გავუშვებთ axis(‘square’) ბრძანების შემდეგ მივიღებთ ნახ 7.14.



ნახ 7.14. სფეროს ზედაპირის კონტური ღერძების კვადრატული თანაფარდობით

პრობლემა: terrain navigation ტოპოგრაფიული გაზომვები?????

terrain navigation ძირითადი კომპონენტია დისტანციურად მართვადი ობიექტების წარმოებაში. ასეთებია მაგალითად რობოტი, უპილოტო თვითმფრინავი და რაკეტა, თვითმართვადი წყალქვეშა ხომალდი და სხვა. ამ მოწყობილობებს ბორტზე გააჩნიათ კომპიუტერი, რომელშიც შეტანილია ინფორმაცია იმ გარემოს შესახებ, სადაც ობიექტი მოძრაობს. ვიცით რა დროის ნებისმიერ მომენტში ობიექტის მდებარეობა შესაძლებელია შეირჩეს დანიშნულების ადგილამდე მისასვლელი ოპტიმალური გზა. როცა დანიშნულების ადგილი შეიცვლება, კომპიუტერი მიმართავს ინფორმაციას გარემოს შესახებ და გამოითვლის ახალ მიმართულებას.

კომპიუტერული პროგრამა, რომელიც ასეთ სისტემებს მართავს, წინასწარ გამოიცდება სხვადასხვაგვარ ტოპოგრაფიულ პირობებში. მონაცემთა ბაზებში არსებობს დედამიწიასა თუ ოკეანის ფსკერის შესახებ ტოპოგრაფიული ინფორმაცია. იმისათვის რომ განისაზღვროს გასავლელი გზის სირთულე საჭიროა განისაზღვროს პიკების რაოდენობა. პიკი ეს არის წერტილი, რომელის სიმაღლეც ყველაზე მეტია მის ირგვლივ წერტილების მიმართ.

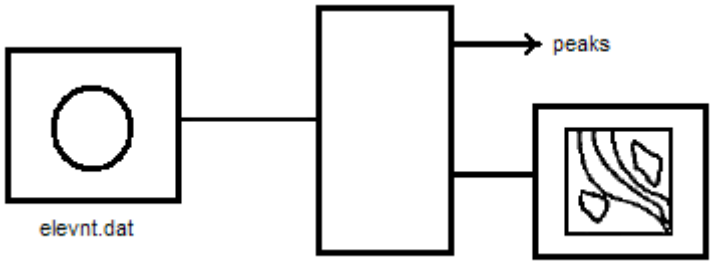
დაწერეთ პროგრამა, რომელიც წაიკითხავს ტოპოგრაფიულ მონაცემებს ფაილიდან და დაბეჭდავს პიკების მდებარეობას საკოორდინატო ბადეზე. ასევე დაბეჭდავს სამგანზომილებიან mesh და კონტურულ გრაფიკს ფაილში არსებული მონაცემების საფუძველზე.

1. ამოცანის დასმა

დგანსაზღვრეთ პიკების რაოდენობა და მდებარეობა ფალში ჩაწერილი მონაცემების საფუძველზე.

2. INPUT/OUTPUT აღწერა

ნახ 7.17 შეიცავს INPUT/OUTPUT დიაგრამას. დიაგრამიდან ჩანს, რომ საწყისი მონაცემები წარმოდგენილია ფაილის სახით, ხოლო შედეგი – პიკების მდებარეობა ცხრილის სახით და მონაცემთა საფუძველზე აგებული კონტურული გრაფიკი.



ნახ 7.15. I/O დიაგრამა

3. სახელდახელო ამოხსნა

დავუშვათ ზედაპირი დაყოფილია 6 × 8 კვადრატად და გაზომილია თითოეული მათგანის სიმაღლე. მონაცემები ჩაწერილია მატრიცის სახით:

25	59	63	23	21	34	21	50
32	45	43	30	37	32	30	27
34	38	38	39	36	28	28	35
40	45	42	48	32	30	27	25
39	39	40	42	48	49	25	30
31	31	31	32	32	33	44	35

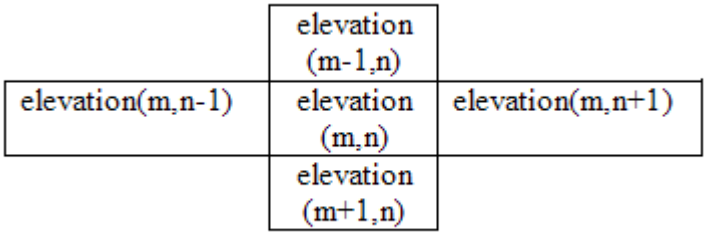
პიკის წერტილების შესაბამისი მნიშვნელობები ხაზგასმულია, ისე, რომ პიკების მდებარეობები შეესაბამება მატრიცის ელემენტებს (2,5), (4,2), (4,4), და (5,6).

4. MATLAB ამოხსნა

პიკების ძიება გულისხმობს ბადის მხოლოდ შიდა წერტილებს. კიდის წერტილი არ შეიძლება ჩაითვალოს პიკად იმიტომ, რომ არ ვიცით მისი ყველა მეზობელი წერტილის მნიშვნელობა. განვიხილოთ სიმაღლე წერტილში, რომელიც შეესაბამება მონაცემს elevation (m,n). მისი მოსაზღვრე მდებარეობები ნაჩვენებია ნახ 7.16 ამდენად, elevant(m,n) პიკი იქნება იმ შემთხვევაში, თუ თუ სრულდება პირობაები:

```
elevation(m,n) > elevation(m,n-1)
elevation(m,n) > elevation(m-1,n)
elevation(m,n) > elevation(m,n+1)
```

```
elevation(m,n) > elevation(m+1,n)
```



ნახ 7.16. elevtion(m,n) მეზობელი წერტილების მდებარეობები

პროგრამის დაწერისას ვიყენებთ ერთმანეთში ჩალაგებული for ციკლის ოპერატორს.

```
%     This program reads the elevation data for
%     a land grid and determines the peaks.
%     It also ganarates a contour plot
%
load elevant.dat
elevation = elevant;
%
%     Identify rhe peaks.
%
[rows, cols] = size(elevation);
for m=2:rows-1
    for n=2:cols-1
        if (elevation(m,n)>elevation(m,n-1)&...
           elevation(m,n)>elevation(m-1,n)&...
           elevation(m,n)>elevation(m,n+1)&...
           elevation(m,n)>elevation(m+1,n))
                fprintf('Peak at (%6.0f, %6.0f) \n',m,n)
         end
    end
end
%
%     Generate plot
%
contour(elevation),...
title('Elevation Data')
```

5. შემოწმება

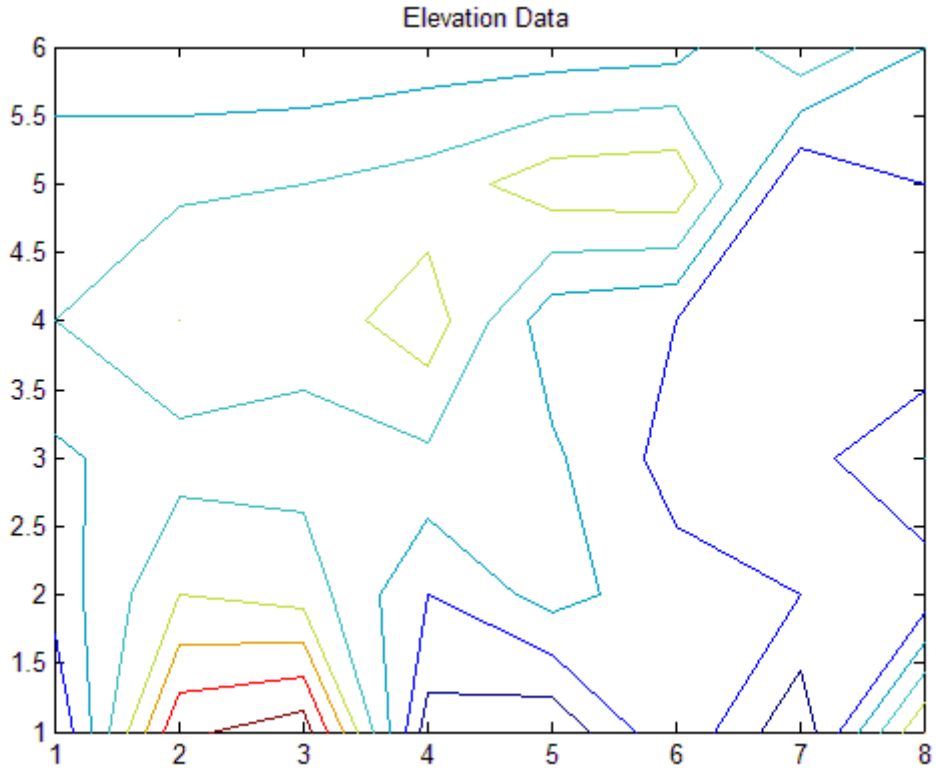
შევამოწმოთ პროგრამა ზემოთ განხილული მონაცემებისათვის, რომელიც ჩაწერილია ფაილში elevant.dat

პიკებისათვის მივიღებთ:

```
Peak at (     2,      5)
```

```
Peak at (     4,      2)
Peak at (     4,      4)
Peak at (     5,      6)
```

როგორც მოსალოდნელი იყო, ხოლო შესაბამისი გრაფიკი მოცემულია ნახ 7.17-ზე.



ნახ 7.17. ტოპოგრაფიულ მონაცემთა კონტურული გრაფიკი

ამ თავში მიმოვიხილეთ MATLAB გრაფიკული ბრძანებები. x-y გრაფიკი, რომელიც ყველაზე ხშირად გამოიყენება. MATLAB საშუალებას იძლევა გამოვიყენოთ მისი როგორც წრფივი, ასევე ლოგარითმული ფორმა. ასევე სასარგებლო და საჭიროა ერთიდაიმავე გრაფიკულ ფანჯარაში რამდენიმე გრაფიკის აგება მათი ერთმანეთთან შედარების თვალსაზრისით.

მრავალ ამოცანაში ფართოდ იყენებენ სამგანზომილებიან მონაცემებს, რომლებიც შეგვიძლია ავაგოთ MATLAB-ში როგორც სამგანზომილებიანი ზედაპირი.

4.8 ბრძანებები და ფიუნქციები

axis	აკონტროლებს ღერძების მასშტაბს
bar	აგებს bar გრაფიკს
clc	ასუფთავებს ბრძანებათა ფანჯარას

clf	ასუფთავებს გრაფიკულ ფანჯარას
contour	აგებს კონტურულ გრაფიკს
ginput	იღებს კოორდინატებს გრაფიკული ფანჯრიდან
grid	გრაფიკს უმატებს საკოორდინატო ბადეს
gtext	საშუალებას გვაძლევს ჩავწეროთ ტექსტი უშუალოდ გრაფიკულ ფანჯარაში
hold	ტოვებს მომდინარე გრაფიკს გრაფიკულ ფანჯარაში
home	ბრძანებების ფანჯარაში კურსორი გადააქვს home მდებარეობაში
loglog	აგებს log-log გრაფიკს
mesh	ააგებს სამგანზომილებიან mesh გრაფიკს
meshgrid	აწარმოებს ვექტორებს საკოორდინატო ბადიათვის
plot	აგებს წრფივ x-y გრაფიკს
polar	აგებს პოლარულ გრაფიკს
print	ბეჭდავს მაღალი გარჩევის გრაფიკს
semilogx	აგებს ლოგარითმულ-წრფივ გრაფიკს
semilogy	აგებს წრფივ-ლოგარითმულ გრაიკს
shg	ეკრანზე გამოყავს გრაფიკული ფანჯარა
stairs	აგებს საფეხუროვან (stair) გრაფიკს
subplot	დაყოფა გრაფიკულ ფანჯარას რამდენიმე ქვეფანჯარად
text	ჩაწერს მითითებულ ტექსტს გრაფიკულ გამონასახში
title	გრაფიკს თავზე წააწერს სათაურს
xlabel	დააწერს მითითებულ აღნიშვნას x ღერძს
ylabel	დააწერს მითითებულ აღნიშვნას y ღერძს
surf	აგებს ზედაპირის ფერად, პარამეტრულ გამოსახულებას
surfc	იგივე surf მაგრამ უმატებს გრაფიკს შესაბამის კონტურულ გამოსახულებას
shading	აკონტროლებს ზედაპირზე ფერების სიგლუვეს(shading)
colormap	განსაზღვრავს ფერთა გამს (დიაპაზონს)

პრობლემები

1-13 პრობლემები დაკავშირებულია ამ თავში განხილულ ამოცანებთან, ხოლო 14 – 22 სხვა ამოცანებს უკავშირდება.

მეტეოროლოგიური რაკეტის ტრაექტორია. ეს ამოცანები დაკავშირებულია ამ თავში განხილულ პრობლემასთან მეტეოროლოგიური რაკეტის (ზონდის) შესახებ.

1. ააგეთ სიჩქარის წრფივ – ლოგარითმული გრაფიკი. შეადარეთ იგი იგიგვე მონაცემების წრფივ გრაფიკს
2. ააგეთ აჩქარების წრფივ – ლოგარითმული გრაფიკი. შეადარეთ იგი იგივე მონაცემების წრფივ გრაფიკს.
3. ააგეთ სიმაღლისა და სიჩქარის გრაფიკი ერთიდაიგივე ღერძების მიმართ, ერთ ნახაზზე, ხედავთ რაიმე ინფორმაციას, რომელიც თვალსაჩინო არ იყო მათი ცალკე ნახაზზე აგების შემთხვევაში?

4. ააგეთ სიმაღლისა და აჩქარების გრაფიკი ერთიდაიგივე ღერძების მიმართ, ერთ ნახაზზე, ხედავთ რაიმე ინფორმაციას, რომელიც თვალსაჩინო არ იყო მათი ცალკე ნახაზზე აგების შემთხვევაში?
5. ააგეთ სიჩქარის და აჩქარების გრაფიკი ერთიდაიგივე ღერძების მიმართ, ერთ ნახაზზე, ხედავთ რაიმე ინფორმაციას, რომელიც თვალსაჩინო არ იყო მათი ცალკე ნახაზზე აგების შემთხვევაში?
6. წააწერეთ ტექსტი რომელიც უჩვენებს პირველი ეტაპის საწვავის ამოწურვის მომენტს სიჩქარისა და აჩქარების გრაფიკზე.
7. აჩქარების გრაფიკზე აღნიშნეთ ის ინტერვალები, როცა იგი გამოწვეულია მხოლოდ გრავიტაციით.

ტოპოგრაფიული (რელიეფური) ნავიგაცია

8. შეცვალე ამ პრობლემასთან დაკვშირებით შედგენილი პროგრამა ისე, რომ რომ დაბეჭდოს პიკების რაოდენობა რელიეფურ ბადეზე (elevation grid)
9. შეცვალე პროგრამა ისე, რომ მან დაბეჭდოს პიკების ნაცვლად (ღრმულების) მეზობელი წერტილების მიმართ ყველაზე ნაკლები სიმაღლის მქონე წერტილების მდებარეობა ბადეზე
10. შეცვალე პროგრამა ისე, რომ მოგვცეს ბადეზე უმაღლესი და უდაბლესი წერილის მდებარეობა და მნიშვნელობა.
11. შეცვალე პროგრამა ისე, რომ მოგვცეს სიმაღლეთა ბადის საშუალო სიმაღლე.
12. დავუშვათ წერტილებს შორის მანძილი ვერტიკალური და ჰორიზონტალური მიმართულებით 100 ფუტია, იპოვეთ ფუტებში გამოსახული ყოველი პიკის მდებარეობა ბადის ზედა მარცხენა კუთხის მიმართ.
13. დავუშვათ წერტილებს შორის მანძილი ვერტიკალური და ჰორიზონტალური მიმართულებით 100 ფუტია, იპოვეთ ფუტებში გამოსახული ყოველი პიკის მდებარეობა ბადის ქვედა მარცხენა კუთხის მიმართ.

ლოგარითმული გრაფიკი. მრავალ გამოყენებით ამოცანაში გვჭირდება ექსპერიმენტული ანათვლების აღება, რომელიც შემდეგ მათემეტიკურად უნდა აღვწეროთ, განვსაზღვროთ განტოლება, რომელიც მიღებულ მონაცემებს ასახავს, ასე ვთქვათ შევქმნათ პროცესის მათემატიკური მოდელი. მაგალითად, თუ ავაგეთ x-y გრაფიკი და ჩანს, რომ იგი ახლოსაა წრფესთან, შეგვიძლია შევაფასოთ მისი დახრის კუთხე და ღერძებთან გადაკვეთის წერტილები და ამის მიხედვით დავწეროთ წრფის განტოლება. განტოლება იქნება მოცემულ მონაცემთა მოდელი.

14. განვიხილოთ შემდეგი მაჩვენებლიანი განტოლება:

$$y = 3 \cdot 10^{2x}$$

თუ x გარკვეულ მნიშვნელობათათვის გამოვითვლით y მნიშვნელობებს და ავაგებთ x-y წრფივ-ლოგარითმულ გრაფიკს, წერტილები წრფეზე განლაგდებიან. ამაში რომ დავრწმუნდეთ ავიღოთ ორივე მხარის ლოგარითმი:

$$\log_{10} y = 2x + \log_{10} 3$$

ეს განტოლება წრფივია x და $\log_{10} y$ -თვის. MATLAB საშუალებით ააგეთ $y = 3 \cdot 10^{2x}$

გამოსახულების წრფივი და წრფივ-ლიგარითმული გრაფიკი. შეაფასეთ დახრა და y ღერძის გადაკვეთის წრეტილი. დახრა დახლოებით 2 უნდა იყოს, ხოლო y ღერძის გადაკვეთის წერტილი $\log_{10} 3$-ის ტოლი.

15. დავუშვათ გაქვთ ექსპერიმენტის შედეგად მიღებულ მონაცემთა მწკრივი, ააგეთ მისი წრფივ-ლოგარითმული გრაფიკი და დაახლოებით წრფე მიიღეთ. რა ტიპის განტოლება შეიძლება განვიხილოთ ასეთი მონაცემების მოდელად? როგორ გამოითვლით განტოლების მუდმივებს?

16. განვიხილოთ შემდეგი განტოლება:

$$y = 5x^3$$

თუ ავაგებთ ამ განტოლების ლოგარითმულ-ლოგარითმულ გრაფიკს, მივიღებთ წრფეს. იმიტომ, რომ თუ ავიღებთ განტოლების ორივე მხარის ლოგარითმს, მივიღებთ:

$$\log_{10} y = 3\log_{10} x + \log_{10} 5$$

ეს განტოლება კი წრფეს წარმოადგენს $\log_{10} x$ და $\log_{10} y$ -თვის. MATLAB საშუალებით ააგეთ $y = 5x^3$ განტოლების წრფივი და ლოგარითმულ-ლოგარითმული გრაფიკი. ამ უკანასკნელის მიხედვით შეაფასეთ დახრა და y ღერძის გადაკვეთის წერტილი. დახრა დაახლოებით 3 –ის ტოლი უნდა მიიღოთ, y ღერძის გადაკვეთის წერტილი კი დაახლოებით $\log_{10} 5$ - ტოლი.

17. დავუშვათ გაქვთ ექსპერიმენტის შედეგად მიღებულ მონაცემთა მწკრივი, ააგეთ მისი ლოგარითმულ-ლოგარითმული გრაფიკი და დაახლოებით წრფე მიიღეთ. რა ტიპის განტოლება შეიძლება განვიხილოთ ასეთი მონაცემების მოდელად? როგორ გამოითვლით განტოლების მუდმივებს?

ფუნქცია sinc . ეს ფუნქცია საინტერესოა თავისი ფორმის გამო. მას აგრეთვე დიდი მნიშვნელობა აქვს ციფრული სიგნალის დამუშავებაში. ფუნქცია აღიწერება ფორმულით:

$$f(x, y) = \sin c(r) = \sin(r)/r$$

სადაც r (x,y) წერტილის მანძილია კოორდინატთა სათავიდან. სათავეში ფუნქციის მნიშვნელობა 1-ის ტოლია (ამ ფუნქციას, ფორმის გამო, ხშირად 'სომბრეროს' უწოდებენ).

18. შექმენით კვადრატული მატრიცა, რომლის ზომა იქნება (11 × 11), რომელიც შეიცავს sinc ფუნქციის მნიშვნელობებს ინტერვალში $-10 \le x \le 10, \quad -10 \le y \le 10$. ააგეთ მიღებული ზედაპირის სამგანზომილებიანი გრაფიკი.
19. მე-18 ამოცანაში განხილული ზედაპირისთვის ააგეთ კონტურული გრაფიკი.
20. ააგეთ მე-18 ამოცანაში განხილული sinc ფუნქციის სამგანზომილებიანი გრაფიკი ხედვის კუთხით ზედაპირის ქვემოდან
21. ააგეთ მე-18 ამოცანაში განხილული sinc ფუნქციის სამგანზომილებიანი გრაფიკი ხედვის კუთხით დახრილობა = 0 გრადუსი.
22. ააგეთ sinc ფუნქციის კონტურული გრაფიკი კონტურებით დონეებზე: 0.1, 0.2, 0.3, 0.4, 05.

ნაწილი III რიცხვითი მეთოდები

ამ ნაწილში წარმოდგენილი რიცხვითი მეთოდები საჭიროა მთელი რიგი პრობლემების გადასაწყვეტად. MATLAB ბრძანებების გარდა ჩავრთეთ რამდნიმე დიაგრამა და გრაფიკი, რომლებიც დაგეხმარებათ გაერკვეთ რიცხვით მეთოდთა კონცეფციის აღქმასა და გააზრებაში. როცა გაიგებთ კონცეფციის არსს, შეძლებთ სწორად შეარჩიოთ მეთოდი დასახული ამოცანის ამოსახსნელად. ასევე ძალზე მნიშვნელოვანია შევამოწმოთ ჩვენს მიერ კომპიუტრის საშუალებით მიღებული შედეგები რამდენად სწორ პასუხს გვაძლევს დასახული პრობლემის ამოხსნაში. რიცხვითი მეთოდებით მიღებული შედედგების ანალიზისა და ინტერპრეტაციისათვის MATLAB-ს მძლავრი გრაფიკული შესაძლებლობები გააჩნია. წფივ განტოლებათა სისტემის ამოხსნა, ინტერპოლაცია, მრუდის გამოთვლა (curve fitting), პოლინომის ფესვების პოვნა – ეს ყველაფერი ხშირად დაგჭირდებათ მრავალი პრობლემის გადასაჭრელად. დანარჩენი თავები – პოლინომური ანალიზი, რიცხვითი ინტეგრება და დიფერენცირება, ჩვეულებრივი დიფერენციალური განტოლება, მატრიცების მამრავლებად დაშლა და სიგნალის დამუშავება უფრო სპეციალიზებულია, ამიტომ გირჩევთ ზოგადად გაეცნოთ რა შესაძლებლობანი გაჩნია MATLAB -ს ამ მიმართულებით და უფრო დაწვრილებით შემდეგ შეისწავლოთ, როცა ამის საჭიროება შეიქმნება.

8 წრფივ განტოლებათა სისტემა

პრობლემა: სატრანსპორტო საშუალებები (Vehicle performance)

General Motors EV1 – ელექტროავტომობილი

სურათზე ხადავთ ჯენერალ მოტორსის მერ წარმოებულ ერთ-ერთ პირველ ავტომობილს. იგი წარმოადგენს სატრანსპორტო საშუალებას გამონაბოლქვის გარეშე. პირველი თაობის ელექტრომობილის ძრავა იკვებებოდა ტყვიის აკუმულატორით (lead-acid battery), რომელიც მეორე თაობის ძრავაში, 1999 წლიდან, შეცვალა ნიკელ-მეტალ-ჰიდრიდულმა აკუმულატორმა. პირველი თაობის ელექტრომობილი ავითარებს სიჩქარეს 90-120კმ/სთ, ხოლო მეორე თაობის ელექტრომობილი – 120-160 კმ/სთ. აკუმულატორის სრული დატვირთვისათვის საჭიროა 8 საათი და მისი სიმძლავრე 18 - 26 კილოვატ/საათს შეადგენს. 1994 წელს მოდიფიცირებულმა ელექტრომობილმა რეკორდული სიჩქარე – 295 კმ/სთ განავითარა. შემდეგი თაობის ელექტრომობილს ექნება უკეთესი მახასიათებლები და კონკურენციას გაუწევს საწვავზე მომუშავე სათრანსპორტო საშუალებებს.

შესავალი

შესავალი

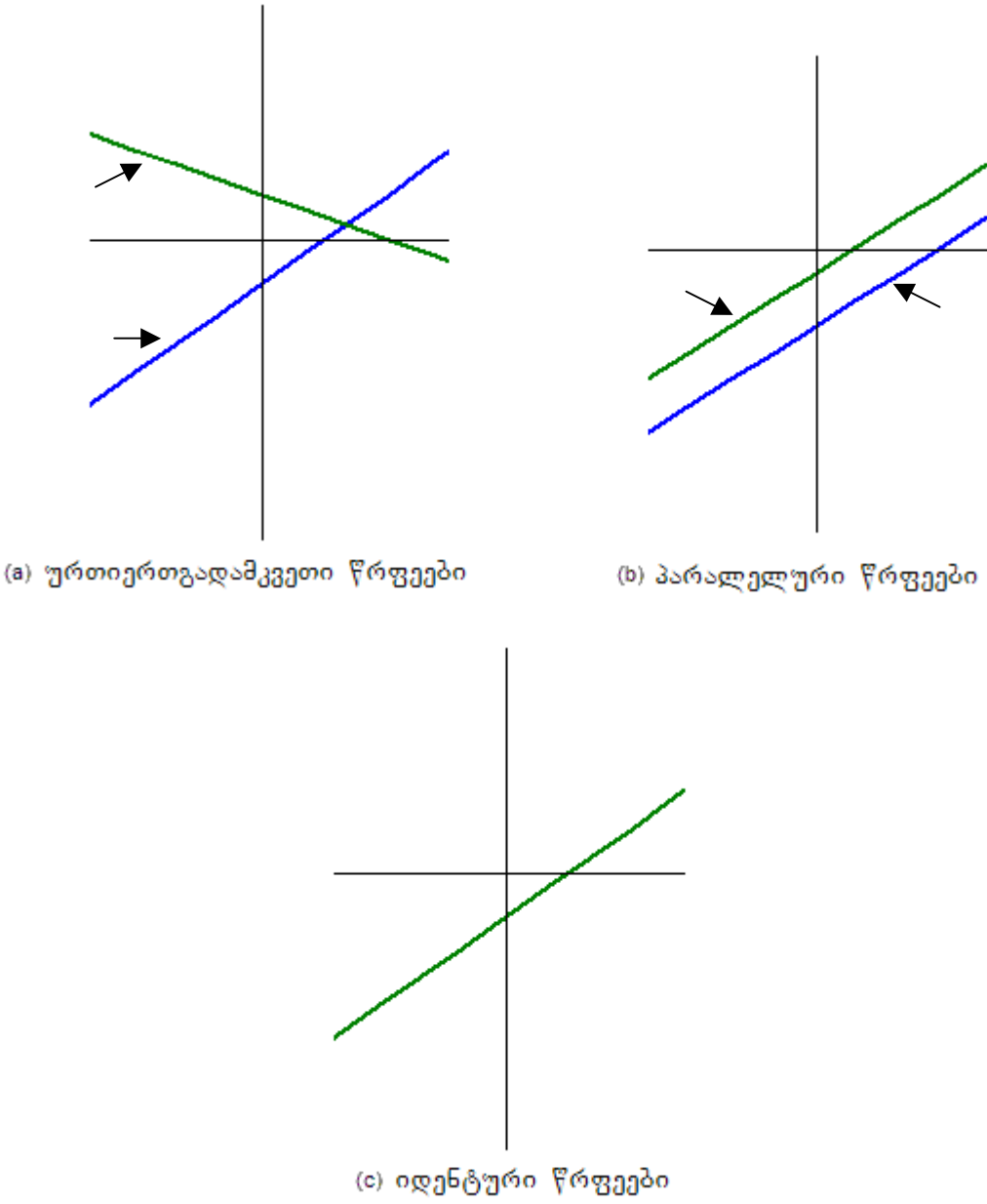
ამ თავს დავიწყებთ წრფივ განტოლებათა სისტემის ამოხსნის გრაფიკული მეთოდის აღწერით. გრაფიკები გვიჩვენებს ორი ან სამუცნობიანი წრფივ განტოლებათა სისტემის ამოხსნის რამდენიმე განსხვავებულ შემთხვევას. სამზე მეტ უცნობიანი სისტემის ამოხსნა განხილულია როგორც ჰიპერსიბრტყეთა თანაკვეთა. წარმოგიდგენთ წრფივ განტოლებათა ამოხსნის ორ სხვადასხვა მეთოდს მატრიცული ოპერაციების საშუალებით. ბოლოს განვიხილავთ მაგალითს – ელექტრული წრედის ანალიზი, სადაც გამოვიყენებთ წრფივ განტოლებათა სისტემის ამოხსნის განხილულ მეთოდებს.

8.1 გრაფიკული ინტერპრეტაცია

არსებობს წრფივ განტოლებათა ამოხსნის მრავალი მეთოდი, მაგრამ თითქმის ყველა მათგანი მოითხოვს მრავალი ოპერაციის განხორციელებას, რომელთა შესრულებისას ადვილდ შეიძლება დავუშვათ შეცდომა. მათ ამოსახსნელად ძალზე მოხერხებულია კომპიუტერის გამოყენება. მაგრამ კარგად უნდა გვესმოდეს ამოხსნის მთელი პროცესი, რომ სწორად შევადგინოთ ალგორითმი და მივიღოთ სწორი შედეგი. პრიცესის კარგად გასაგებად დავიწყოთ წრფივ განტოლებათა სისტემის ამოხსნის გრაფიკული ინტერპრეტაციით.

წრფივი განტოლება ორი უცნობით როგორიცაა $2x - y = 3$ წარმოადგენს წრფეს და ხშირად ასეთი ფორმით წერენ: $y = mx + b$, სადაც m წრფის დახრაა და b - y ღერძის გადაკვეთის წერტილი. თუ გვაქვს 2 წრფივი განტოლება, გვაქვს ორი წრფე, რომლებიც ან ერთ წერტილში იკვეთება, ან ურთიერთპარალელურია ან ერთიდაიგივე წრფეს წარმოადგენს. სამივე შემთხვევა ნაჩვენებია ნახ 8.1. განტოლებებს, რომლებიც ურთიერთგადამკვეთ წრფეებს წარმოადგენს, ადვილად გამოვიცნობთ იმით, რომ მათ განსხვავებული დახრა აქვთ. მაგალითად: $y = 2x - 3$ და $y = -x + 3$. განტოლებებს, რომლებიც ორ ურთიერთპარალელურ წრფეს წარმოადგენს აქვთ ერთნაირი დახრა მაგრამ განსხვავებული y ღერძთან გადაკვეთის წერტილი: $y = 2x - 3$ და $y = 2x + 1$. განტოლებებს, რომლებიც ერთიდაიმავე წრფეს

წარმოადგენს აქვთ ერთნაირი დახრა და y ღერძთან გადაკვეთის ერთიდაიგივე წერტილი: $y = 2x - 3$ და $3y = 6x - 9$.

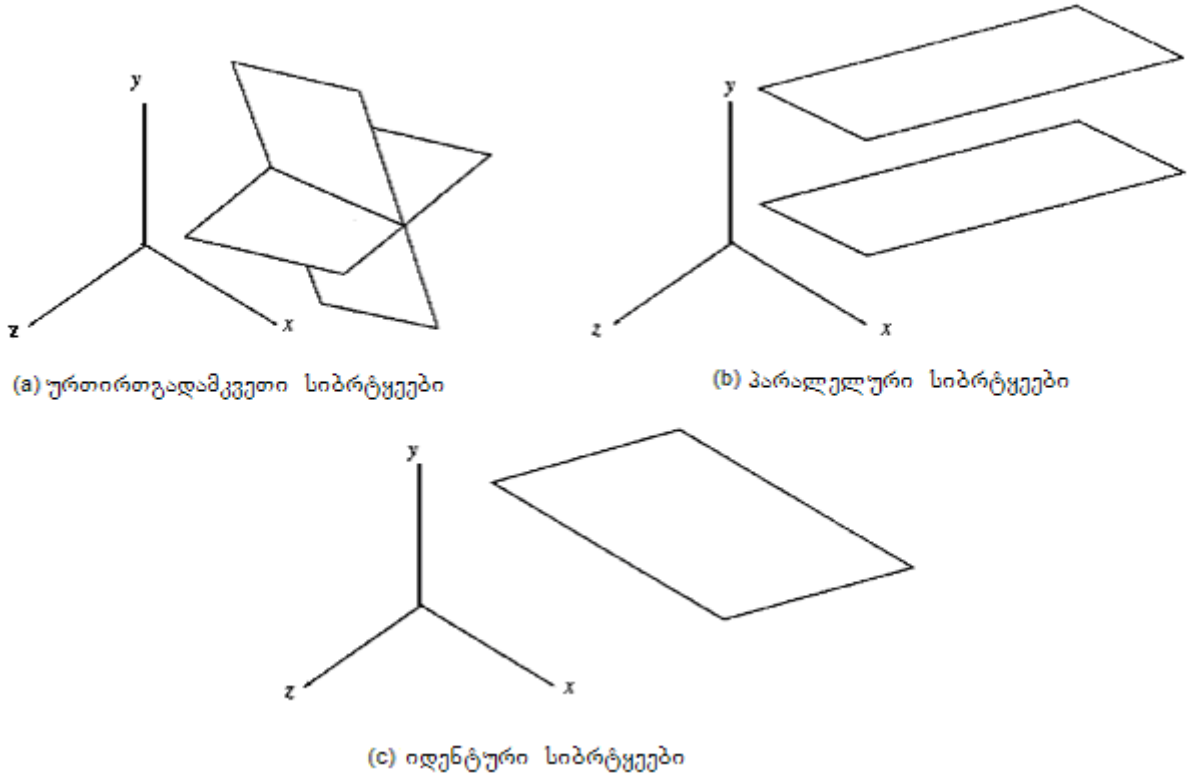


ნახ 8.1. ორი წრფე

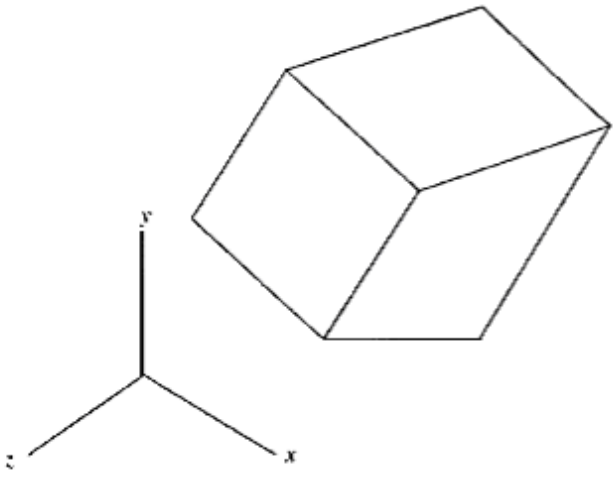
თუ განტოლება შეიცავს სამ ცვლადს, x, y, z, მაშინ ის წარმოადგენს სიბრტყეს სამგანზომილებიან სივრცეში. თუ გვაქვს ორი განტოლება სამი უცნობით ისინი შეიძლება წარმოადგენდეს ორ სიბრტყეს, რომლებიც იკვეთება წრფეზე, ორ პარალელურ სიბრტყეს, ან ორ ერთიდაიგივე სიბრტყეს. ეს შენთხვევები ნაჩვენებია ნახ 8.2-ზე. თუ გვაქვს სამი განტოლება სამი ცვლადით, მაშინ ეს სამი სიბრტყე შეიძლება იკვეთებოდეს ერთ წერტილში, სამ პარალელურ წრფეზე, ორ პარალელურ წრფეზე და ერთ წრფეზე. შეიძლება არსად არ იკვეთებოდნენ, იყვნენ ურთიერთპარალელური, ანდა სამივე ერთიდაიგივე სიბრტყეს წარმოადგენდეს. (ნახ 8.3, ნახ 8.4).

მსჯელობა შეიძლება განვავრცოთ სამზე მეტი ცვლადის შემცველ განტოლებაზე, მაგრამ ძნელია მათთვის შესაბამისი ვიზუალიზაციის განხორციელება. წერტილთა ერთობლიობას, რომელიც განისაზღვრება სამზე მეტი ცვლადის მქონე წრფივი გატოლებით ჰიპერსიბრტყეს ვუწოდებთ. საზოგადოდ შეგვიძლია განვიხილოთ M წრფივ განტოლებათა სისტემა N ცვლადით, სადაც თითოეული განტოლება განსაზღვრავს კერძო ჰიპერსიბრტყეს, რომელიც ამ სისტემის არცერთი სხვა ჰიპერსიბრტყის იდენტური არ არის. თუ M < N, სისტემა განუზღვრელია და არ არსებობს მისი ამოხსნა. თუ M = N, სისტემას აქვს ერთადერთი ამოხსნა იმ შემთხვევაში, თუ სისტემა არ შეიცავს პარალელურ ჰიპერსიბრტყეებს. თუ M > N, სისტემა (overspecified) და მას არ აქვს ერთადერთი ამოხსნა. განტოლებათა სისტემას, რომელსაც გააჩნია ერთადერთი ამოხსნა, არასინგულარული ეწოდება, ხოლო თუ არ გააჩნია – სინგულარული (განსაკუთრებული).

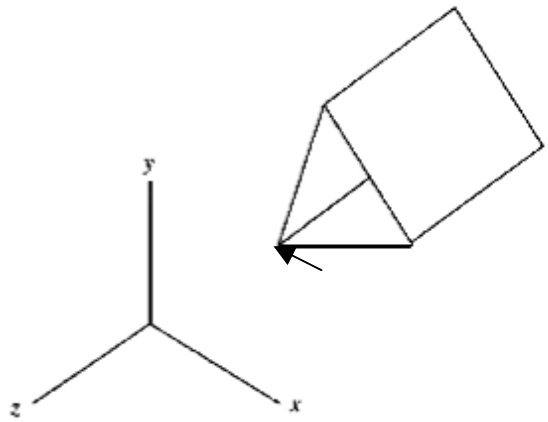
ხშირად გვჭირდება განვსაზღვროთ აქვს თუ არა განტოლებათა სისტემას ამონახსნი. შემდეგ განყოფილებაში განვიხილავთ MATLAB საშუალებით წრფივ გამტოლებათა სისტემის ამოხსნის ორ სხვადასხვა ხერხს.



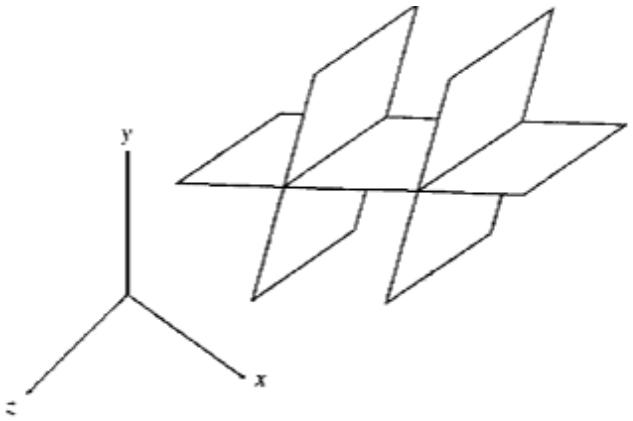
ნახ 8.2. ორი სიბრტყე



ერთ წერტილში თანამკვეთი სამი სიბრტყე

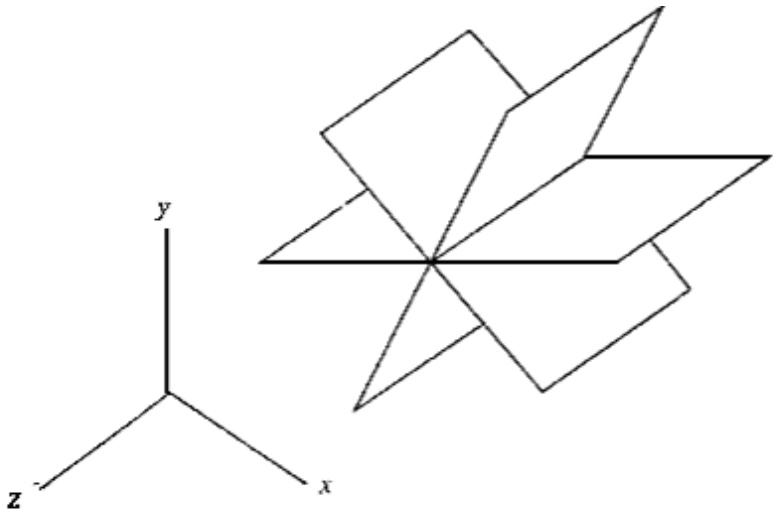


სამ სიბრტყეს გააჩნია თანაკვეთის 3 პარალელური წრფე

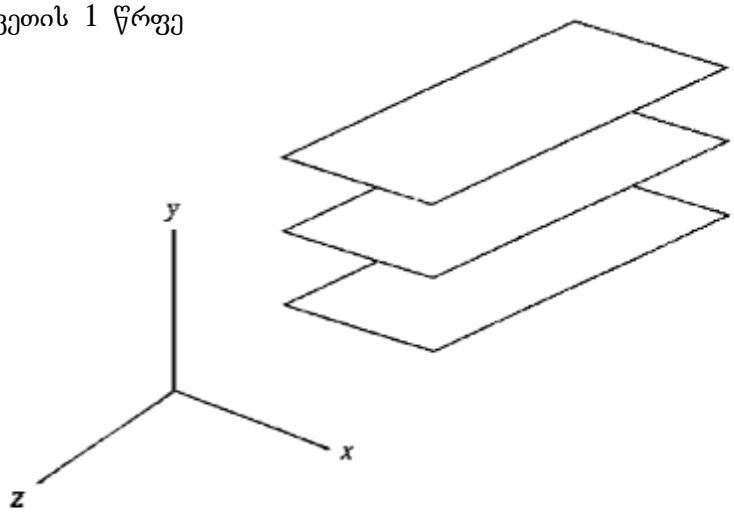


სამ სიბრტყეს გააჩნია თანაკვეთის 2 პარალელური წრფე

ნახ 8.3. სამი სიბრტყე



სამ სიბრტყეს გააჩნია თანაკვეთის 1 წრფე



სამი პარალელური სიბრტყე

ნახ 8.4. სამი სიბრტყე (გაგრძელება)

8.2 ამოხსნა მატრიცული ოპერაციებით

განვიხილოთ განტოლებათა სამუცნობიანი სისტემა:

$$\begin{bmatrix} 3x+2y-z=1 \\ -x+3y+2z=5 \\ x-y-z=-1 \end{bmatrix}$$

სისტემა შეგვიძლა ჩავწეროთ მატრიცების საშუალებით:

$$A=\begin{bmatrix} 3 & 2 & -1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{bmatrix} \qquad X=\begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad B=\begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix}$$

თუ გავიხსენებთ მატრიცების გამრავლების წესს, განტოლებათა ეს სისტემა ასე შეგვიძლია ჩავწეროთ: შეასრულეთ მატრიცების გადამრავლების ოპერაცია, რომ დარწმუნდეთ ამაში.

როცა ცვლადებს სხვადასხვა ასოთი აღვნიშნავთ, სისტემის ჩაწერა მოუხერხებელია, ამიტომ აღვნიშნოთ ცვლადები ერთიდაიგივე ასოთი, განსხვავებული ინდექსით: x_1, x_2, x_3 და ა. შ. მივიღებთ ასეთი სახის სისტემას:

$$\begin{bmatrix} 3x_1 + 2x_2 - x_3 = 1 \\ -x_1 + 3x_2 + 2x_3 = 5 \\ x_1 - x_2 - x_3 = -1 \end{bmatrix}$$

ეს სისტემა მატრიცულად ასე ჩაიწერება: $AX=B$, სადაც X არის სვეტი ვექტორი

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$$

შეგვიძლია სისტემა მატრიცულად სხვა ფორმითაც ჩავწეროთ: $XA=B$, სადაც:

$$X = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \quad A = \begin{bmatrix} 3 & -1 & 1 \\ 2 & 3 & -1 \\ -1 & 2 & -1 \end{bmatrix} \quad B = \begin{bmatrix} 10 & 5 & -1 \end{bmatrix}$$

გადაამრავლეთ მატრიცები და დარწმუნდით მსჯელობის სისწორეში. (შევნიშნავთ, რომ მატრიცა A ამ სისტემაში საწყისი სისტემის A მატრიცის ტრანსპონირებულია).

ახლა განვიხილოთ N უცნობიან N განტოლებათა სისტემის ამოხსნის ორი ხერხი. თუმცა MATLAB-ში შესაძლებელია განუზღვრელი სისტემის ამოხსნაც, სადაც შედეგი წარმოადგენს უმცირეს კვადრატულ ამონახსნს (list square solution) და აქ არ განვიხილავთ. დავუშვებთ, რომ გვაქვს N განტოლება N უცნობით.

8.2.1 მატრიცების გაყოფა

MATLAB –ში წფივ განტოლებათა სისტემა შეგვიძლია ამოვხსნათ მატრიცების გაყოფით. $AX=B$ სისტემის ამოხსნა ხორციელდება მატრიცების მარცხენა გაყოფით: X=A\B; XA=B სისტემის ამოხსნა გამოითვლება მატრიცების მარჯვენე გაყოფით: X=B/A. (MATLAB მატრიცების გაყოფისათვის იყენებს გაუსის გამორიცხვის(elimination) რიცხვით მეთოდს).

მაგალითად, ამოვხსნათ განტოლებათა ზემოხსენებული სისტემა. MATLAB საშუალებით ასე განხორციელდება:

```
A = [3,2,-1; -1,3,2; 1,-1,-1];
B = [10,5,-1]';
X = A\B;
```

ვექტორი X შეიცავს სიდიდეებს: -2, 5, -6. შეგვიძლია შევამოწმოთ ამოხსნა, გავამრავლოთ A*X. შედეგად მივიღებთ სვეტ ვექტორს ელემენტებით: 10,5,-1.

იგივე სისტემა შეგვიძლია ამოვხსნათ მატრიცული განტოლებით XA = B შემდეგი ბრძანებებით:

```
A = [3,-1,1; 2,3,-1; -1,2,-1];
B = [10,5,-1];
X = B/A;
```

X ვექტორი შეიცავს –2, 5, -6. შევამოწმოთ ამონახსნი გავამრავლოთ X ვექტორი A ვექტორზე: X*A მივიღებთ სტრიქონ ვექტორს, რომელიც შეიცავს 10, 5, -1.

თუ გატოლებათა სისტემა სინგულარულია, MATLAB მიგვითითებს შეცდომაზე. ამონახსნი ვექტორი შეიძლება შეიცავდეს მნიშვნელობებს – NAN , $+\infty$ ან $-\infty$. ასევე შესაძლებელია სისტემა შეიცავდეს ისეთ განტოლებებს, რომელთა შესაბამისი ჰიპერსიბრტყეები ძალიან ახლოსაა ერთმანეთთან ან ძალიან მცირედ განსხვავდება პარალელურისაგან. ასეთ სისტემებს ill-conditioned სისტემას უწოდებენ. MATLAB გამოითვლის ამოხსნას, მაგრამ მიგვითითებს, რომ შედეგი მაინცდამაინც სანდო არ არის.

8.2.2 მატრიცის შებრუნებული

განტოლებათა ისტემა შეიძლება ამოიხსნას მატრიცების შებრუნების მეთოდით. მაგალითად, დავუშვათ:

$$A=\begin{bmatrix} 3 & 2 & -1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{bmatrix} \qquad X=\begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad B=\begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix}$$

მაშინ AX =B. დავუშვათ ტოლობის ორივე მხარე გადავამრავლოთ A^{-1} ,

$$A^{-1}AX = A^{-1}B$$

მაგრამ რადგანაც $A^{-1}A$ ტოლია ერთეულოვანი მატრიცის - I, გვექნება:

$$IX = A^{-1}B \qquad \text{ანუ} \quad X = A^{-1}B$$

MATLAB –ში ეს ასე შეგვიძლია გამოვთვალოთ:

```
X = inv(A)*B
```

თუმცა ეს მეთოდი განსხვავდება მატრიცული მარცხენა გაყოფის მეთოდისაგან, მაგრამ იგივე შედეგს იძლევა ისეთი სისტემებისათვის, რომლებიც არ არის ill conditioned (слабо обусловленная матрица) სუსტად განპირობებული.

იგივე სისტემა ასევე ამოიხსნება მატრიცების შებრუნების მეთოდით, თუ იგი ჩაწერილია ფორმით XA=B სადაც

$$X = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \quad A = \begin{bmatrix} 3 & -1 & 1 \\ 2 & 3 & -1 \\ -1 & 2 & -1 \end{bmatrix} \quad B = \begin{bmatrix} 10 & 5 & -1 \end{bmatrix}$$

ტოლობის ორივე მხარე გადავამრავლოთ A^{-1} ,

$$XA^{-1}A = BA^{-1}$$

მაგრამ რადგანაც $A^{-1}A$ ტოლია ერთეულოვანი მატრიცის - I, გვექნება:

$$XI = BA^{-1} \qquad \text{ანუ} \quad X = BA^{-1}$$

MATLAB –ში ეს ასე შეგვიძლია გამოვთვალოთ:

```
X = B*inv(A)
```

ყურადღება მიაქციეთ, როცა ამ მეთოდს იყენებთ, რომ B იყოს სტრიქონი ვექტორი.

სავარჯიშო

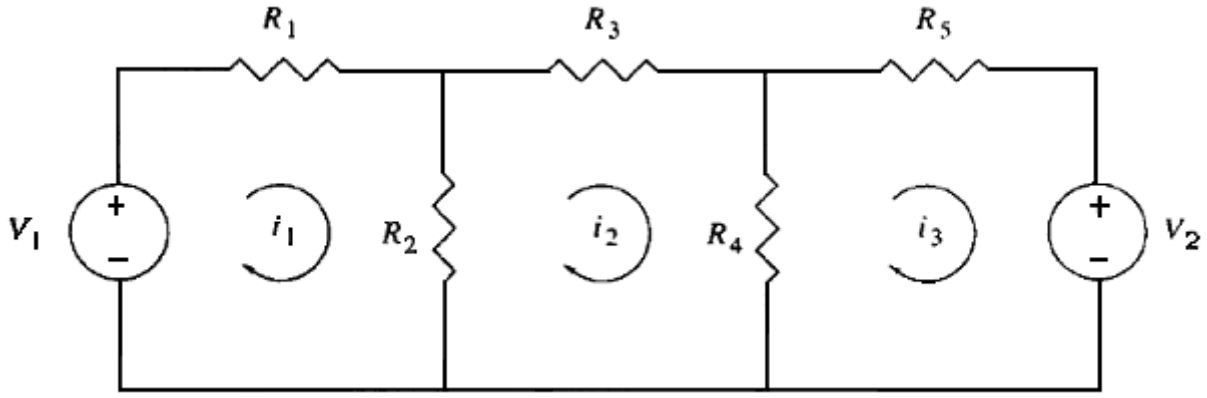
ამოხსენით წრფივ განტოლებათა სისტემა მატრიცების გაყოფის და მატრიცების შებრუნებულის მეთოდით. MATLAB საშუალებით შეამოწმე მიღებული ამონახსნი მატრიცების გამრავლების საშუალებით. ორუცნობიან განტოლებათა სისტემისათვის ააგეთ განტოლებათა შესაბამისი გრაფიკები ერთიდაიმავე ნახაზზე, იმისათვის, რომ უჩვენოთ გადაკვეთა. თუ სისტემას არ აქვს კერძო(unique) ამოხსნა უჩვენეთეს გრაფიკულად.

1. $\begin{bmatrix} -2x_1 + x_2 = -3 \\ x_1 + x_2 = 3 \end{bmatrix}$
2. $\begin{bmatrix} -2x_1 + x_2 = -3 \\ -2x_1 + x_2 = 1 \end{bmatrix}$
3. $\begin{bmatrix} -2x_1 + x_2 = -3 \\ -6x_1 + 3x_2 = -9 \end{bmatrix}$
4. $\begin{bmatrix} -2x_1 + x_2 = -3 \\ -2x_1 + x_2 = -3.00001 \end{bmatrix}$
5. $\begin{bmatrix} 3x_1 + 2x_2 - x_3 = 10 \\ -x_1 + 3x_2 + 2x_3 = 5 \\ x_1 - x_2 - x_3 = -1 \end{bmatrix}$
6. $\begin{bmatrix} 3x_1 + 2x_2 - x_3 = 1 \\ -x_1 + 3x_2 + 2x_3 = 1 \\ x_1 - x_2 - x_3 = 1 \end{bmatrix}$
7. $\begin{bmatrix} 10x_1 - 7x_2 + 0x_3 = 7 \\ -3x_1 + 2x_2 + 6x_3 = 4 \\ 5x_1 + x_2 + 5x_3 = 6 \end{bmatrix}$

8. $$\begin{bmatrix} x_1 + 4x_2 - x_3 + x_4 = 2 \\ 2x_1 + 7x_2 + x_3 - 2x_4 = 16 \\ x_1 + 4x_2 - x_3 + 2x_4 = 1 \\ 3x_1 - 10x_2 - 2x_3 + 5x_4 = -15 \end{bmatrix}$$

პრობლემა: ელექტრული წრედის ანალიზი

ელექტრული წრედის სიხშირული ანალიზისათვის საჭიროა ამოიხსნას წრფივ განტოლებათა სისტემა. ეს განტოლებები მიიღება წრედების თეორიიდან და ასახავენ წრედში დენისა და ძაბვის ხასიათსა და განაწილებას. მაგალითად განვიხილოთ ნახ 8.5-ზე ნაჩვენები ელექტრული სქემა.



ნახ 8.5. წრედი ძაბვის ორი წყაროთი

შეიძლება შევადგინოთ სამი განტოლება, რომლებიც აღწერს ძაბვას სამ სხვადასხვა კონტურში:

$$\begin{aligned} -V_1 + R_1 i_1 + R_2(i_1 - i_2) &= 0 \\ R_2(i_2 - i_1) + R_3 i_2 + R_4(i_2 - i_3) &= 0 \\ R_4(i_3 - i_2) + R_5 i_3 + V_2 &= 0 \end{aligned}$$

თუ დავუშვებთ, რომ წინაღობები (R_1, R_2, R_3, R_4, R_5) და ძაბვები (V_1, V_2) ცნობილია, ამ სისტემაში გვექნება სამი უცნობი (i_1, i_2, i_3). სისტემა შეგვიძლია ჩავწეროთ შემდეგი ფორმით:

$$\begin{bmatrix} (R_1 + R_2)i_1 & -R_2 i_2 & +0i_3 & = V_1 \\ -R_2 i_1 & (R_2 + R_3 + R_4)i_2 & -R_4 i_3 & = 0 \\ +0i_1 & R_4 i_2 & (R_4 + R_5)i_3 & = -V_2 \end{bmatrix}$$

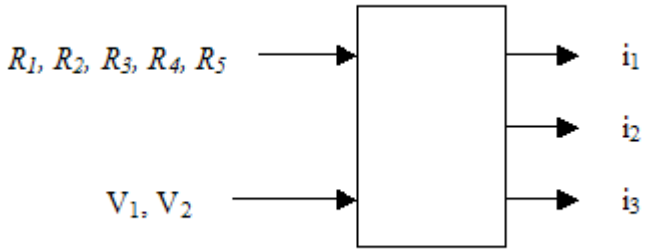
დაწერეთ პროგრამა MATLAB-ში, რომელიც საშუალებას მოგვცემს შევიტანოთ მონაცემები წინაღობებისა და ძაბვებისათვის და მივიღოთ დენის შესაბამისი მნიშვნელობები.

1. **ამოცანის დასმა**

წინაღობების და ძაბვების ცნობილ მნიშვნელობათა საშუალებით ვიპოვოთ დენის მნიშვნელობები ელექტრულ წრედში, რომლის სქემაც ნაჩვენებია ნახ. 8.4.

2. INPUT/OUTPUT აღწერა

ნახ 8.6 შეიცავს დიაგრამას, სადაც ნაჩვენებია რომ გვაქვს ძაბვის 2 და წინაღობის 5 მნიშვნელობა, რომელთა საშუალებით წრფივ განტოლებათა სამუცნობიანი სისტემის საფუძველზე უნდა მივიღოთ დენის ძალის 3 მნიშვნელობა:



ნახ 8.6. Input-Output დიაგრამა

წრფივ განტოლებათა სისტემას ასეთი სახე ექნება:

$$\begin{bmatrix} 2i_1 & -i_2 & +0i_3 & =5 \\ -i_1 & +3i_2 & -i_3 & =0 \\ +0i_1 & -1i_2 & +2i_3 & =-5 \end{bmatrix}$$

გამოვიყენოთ MATLAB განტოლებათა ამ სისტემის ამოსახსნელად:

```
A = [2, -1, 0; -1, 3, -1; 0, -1, 2];
B = [5, 0, -5]';
X = A\B
err = sum(A*X-B)
```

მივიღებთ მნიშვნელობებს;

სამი პარალელური სიბრტყე

```
X =

    2.5000
         0
   -2.5000
```

გარდა ამისა, მივიღებთ A*X-B სიდიდეთა ჯამის მნიშვნელობას, რომელიც ნულის ტოლი უნდა იყოს, თუ სისტემა არასინგულარულია. მართლაც MATLAB მოგვცემს:

```
err =

     0
```

3. MATLAB ამოხსნა

პროგრამა ისე უნდა შევადგინოთ, რომ ბრძანებათა ფანჯარაში დაისვას კითხვა, რომელიც მოგვთხოვს შევიტანოთ ძაბვებისა და წინაღობების მნიშვნელობები. ყურადღება მივაქციოთ ფაქტს, რომ თუ I გამოვიყენებთ, როგორც დენის ძალის აღნიშვნას, იგი დაკარგავს თავის MATLAB-ისეულ მნიშვნელობას $0 + 1.0000\mathrm{i}$.

```
%     This program reads resistor and voltage values
%     and then comlutes the corresponding mesh
%     currents for a specified electrical circuit

R = input('Enter resistor values in ohms, [R1...R5]');
V = input('Enter voltage values in volts, [V1 V2]');
%
%     Initialize matrix A and vector B using AX = B form.


A = [R(1)+R(2),            -R(2),            0;
       -R(2),   R(2)+R(3)+R(4),        -R(4);
          0 ,            -R(4),   R(4)+R(5) ];
       B = [V(1);
              0 ;
            -V(2)];
%
fprintf('Mesh Currents \n')
i = A\B
```

4. შემოწმება

თუ მივაწოდებთ შემდეგ მნიშვნელობებს:

```
Enter resistor values in ohms, [R1...R5]  [1 1 1 1 1]
Enter voltage values in volts, [V1 V2]  [5 5]
```

პროგრამა შედეგად მიგვცემს:

```
Mesh Currents

i =
    2.5000
         0
   -2.5000
```

შეამოწმეთ მიღებული შედეგი A მატრიცის გადამრავლებით i ვექტორზე.

დასკვნა

ეს თავი დავიწყეთ წრფივ განტოლებათა სისტემის გრაფიკული ინტერპრეტაციით. გიჩვენეთ სხვადასხვა შემთხვევა, რომელსაც შეიძლება ადგილი ჰქონდეს წრფივ განტოლებათა ორუცნობიანი და სამუცნობიანი სისტემებისათვის. ეს მსჯელობა შემდეგ განვავრცეთ N

უცნობიანი N განტოლების შემცველ სისტემებზე, იმ დაშვებით, რომ ეს განტოლებები წარმოადგენს განსხვავებულ ჰიპერსიბრტყეებს. განვიხილეთ წრფივ განტოლებათა სისტემის ამოხსნის ორი მეთოდი, რომელიც მატრიცულ ოპერაციებს ეყრდნობა.

\	მატრიცების მარცხენა გაყოფა
/	მატრიცების მარჯვენა გაყოფა

ამოცანები

1-3 პრობლემა უკავშირდება ამ თავში განხილულ ამოცანებს, ხოლო 4 – 8 დაკავშირებულია სხვა ამცანებთან.

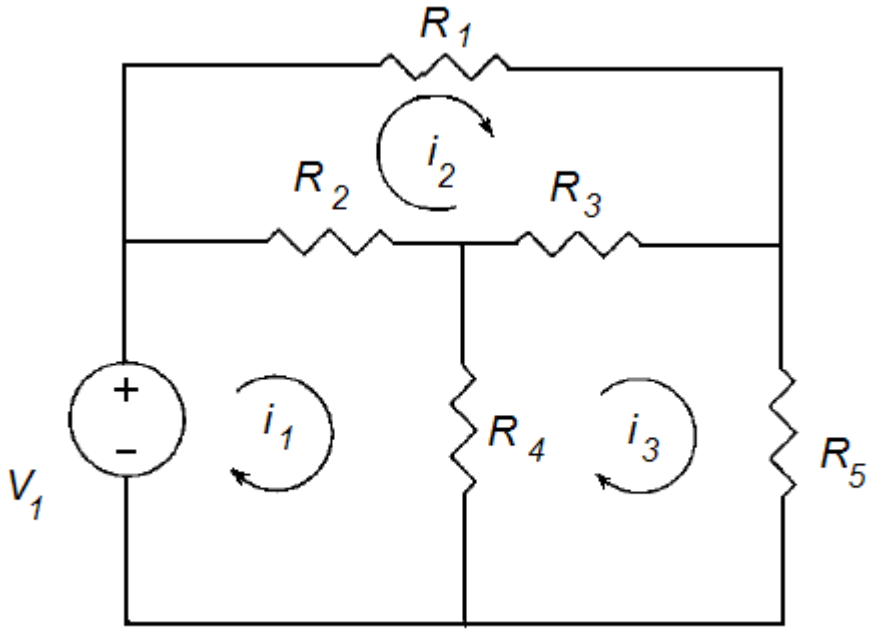
ელექტრული წრედი. ეს ამოცანები დაკავშირებულია ამთავში განხილულ ამოცანასთან ელექტრული წრედის ანალიზის შესახებ.

1. შეცვალე პროგრამა ისე, რომ იგი მოითხოვდეს წინაღობის მნიშვნელობებს კილოომებში, შესაბამისად შეცვალე პროგრამის დანარჩენი ნაწილიც.
2. შეცვალე პროგრამა ისე, რომ ძაბვის ორივე წყაროს ერთიდაიგივე მნიშვნელობა ჰქონდეს ყოველთვის.
3. შეცვალე პროგრამა ისე, რომ ძაბვის ორივე წყაროს მნიშვნელობა იყოს 5 ვოლტი, შემდეგ ჩათვალეთ, რომ წინაღობათა მნიშვნელობები ერთმანეთის ტოლია. გამოითვალეთ დენის მნიშვნელობები წინაღობათათვის: 100, 200, 300, . . . , 1000 ომი.

ელექტრული წრედი ძაბვის ერთი წყაროთი. მოცემულია ელექტრული წრედი ძაბვის ერთი წყაროთი და 5 წინაღობით.

4. გვაქვს განტოლებათა სისტემა ასეთი წრედისათვის, რომლის სქემა მოცემულია ნახ 8.7. დაწერეთ პროგრამა, რომელიც გამოითვლის დენის მნიშვნელობებს ჩვენს მიერ მიწოდებული წინაღობების და ძაბვების მნიშვნელობათათვის:

$$\begin{bmatrix} -V_1 & +R_2(i_1-i_2) & +R_4(i_1-i_3) & =0 \\ R_1 i_2 & +R_2(i_2-i_3) & +R_2(i_2-i_1) & =0 \\ R_3(i_3-i_2) & +R_5 i_3 & +R_4(i_3-i_1) & =0 \end{bmatrix}$$



ნახ 8.7. ელექტრული წრედი ძაბვის ერთი წყაროთი

5. შეცვალე 4 ამოცანის პროგრამა ისე, რომ მან დაბეჭდოს წრფივ განტოლებათა მოცემული სისტემის კოეფიციენტები და მუდმივები

განტოლებათა სისტემა. დავუშვათ გვაქვს მონაცემთა ფაილი eqns.dat, რომელიც შეიცავს წრფივ განტოლებათა სისტემის კოეფიციენტებს. თითოეული სტრიქონი შეიცავს კოეფიციენტებსა და მუდმივს ერთი განტოლებისათვის. მონაცემთა ფაილი შეიცავს განტოლებას N უცნობისათვის.

6. დაწერეთ პროგრამა, რომელიც კითხულობს ფაილს eqns.dat, განსაზღვრავს ხომ არ არის სისტემაში პარალელური ჰიპერსიბრტყეები. (შეგახენებთ, რომ ორ პარალელურ ჰიპერსიბრტყეს აქვს ერთნაირი კოეფიციენტები, მაგრამ განსხვავებული მუდმივები.) დაბეჭდეთ მონაცემები, რომლებიც შეესაბამება პარალელურ ჰიპერსიბრტყეებს.
7. დაწერეთ პროგრამა, რომელიც კითხულობს ფაილს eqns.dat, განსაზღვრავს ხომ არ არის სისტემაში ერთიდაიგივე (ტოლი) ჰიპერსიბრტყეები. (შეგახენებთ, რომ ორ ტოლ ჰიპერსიბრტყეს აქვს ერთნაირი კოეფიციენტები და მუდმივები, ან მიიღებიან ერთმანეთისგან წრფივი გარდაქმნის შედეგად). დაბეჭდეთ მონაცემები, რომლებიც შეესაბამება იდენტურ ჰიპერსიბრტყეებს.
8. გააერთიანეთ ორი წინა ამოცანის პირობა და დაბეჭდეთ მხოლოდ განსხვავებული და არაპარალელური ჰიპერსიბრტყეების შესაბამისი მონაცემები.

ამოხსნა მატრიცის შებრუნებულის მეთოდით. ისარგებლეთ მატრიცის შებრუნებულით, რომ ამოხსნათ წრფივ განტოლებათა შემდეგი სისტემა. ჩაწერეთ სისტემა ორივე ფორმით: A*X=B, X*A=B, შეადარეთ ორივე შემთხვევაში მიღებული შედეგები.

9. $x + y + z + t = 4$
   $2x – y + t = 2$

$3x + y - z - t = 2$
$x - 2y - 3z + t = -3$

10. $2x + 3y + z + t = 1$
    $x - y - z + t = 1$
    $3x + y + z + 2t = 0$
    $-x + z - t = -2$
11. $x - 2y + z + t = 3$
    $x + z = t$
    $2y - z = t$
    $x + 4y + 2z - t = 1$
12. $x + 2y + w = 0$
    $3x + y + 4t + 2w = 3$
    $2x - 3y - z + 5w = 1$
    $x + 2z + 2w = -1$

ურთიერთგადამკვეთი ჰიპერსიბრტყეები. თითოეული მოცემული წერტილისათვის შექმენით წრფივ განტოლებათა ორი განსხვავებული სისტემა, რომლებიც გადაიკვეთება მოცემულ წერტილში.

13. [3, -5, 7]
14. [0, -2, 1.5, 5]
15. [1, 2, 3, -2, -1]

Fermi – კოსმოსური გამა ტელესკოპი

4 ინტერპოლირება და რეგრესია

პრობლემა: ბირთვული სინთეზი

ექსპერიმენტული მონაცემების დამუშავებას მნიშვნელოვანი როლი აკისრია ახალი მეცნიერული პრინციპების და თეორიის შემუშავებაში. ბირთვული სინთეზის პრობლემა მოითხოვს ღრმა ცოდნას ისეთ სფეროში როგორიცაა ბირთვული მდგრადობა და ბირთვული დაშლა. ბირთვული ენერგიის პრობლემები ასოცირდება გამა გამოსხივებასთან. იმისათვის რომ გაიზომოს კოსმოსიდან მოსული გამა გამოსხივება, შეიქმნა გამა გამოსხივების კოსმოსური ობსერვატორია, რომელიც კოსმოსში გაუშვეს 1991 წელს. იგი ორბიტაზე გაუშვა შატლმა ATLANTIS. იგი შეცვალა უფრო მძლავრმა აპარატურამ, რომელიც გაშვებული იქნა 2008 წლის 11 ივნისს. დღეს იგი ცნობილია როგორც კოსმოსური გამა ტელესკოპი **Fermi (http://fermi.gsfc.nasa.gov/)**.

შესავალი

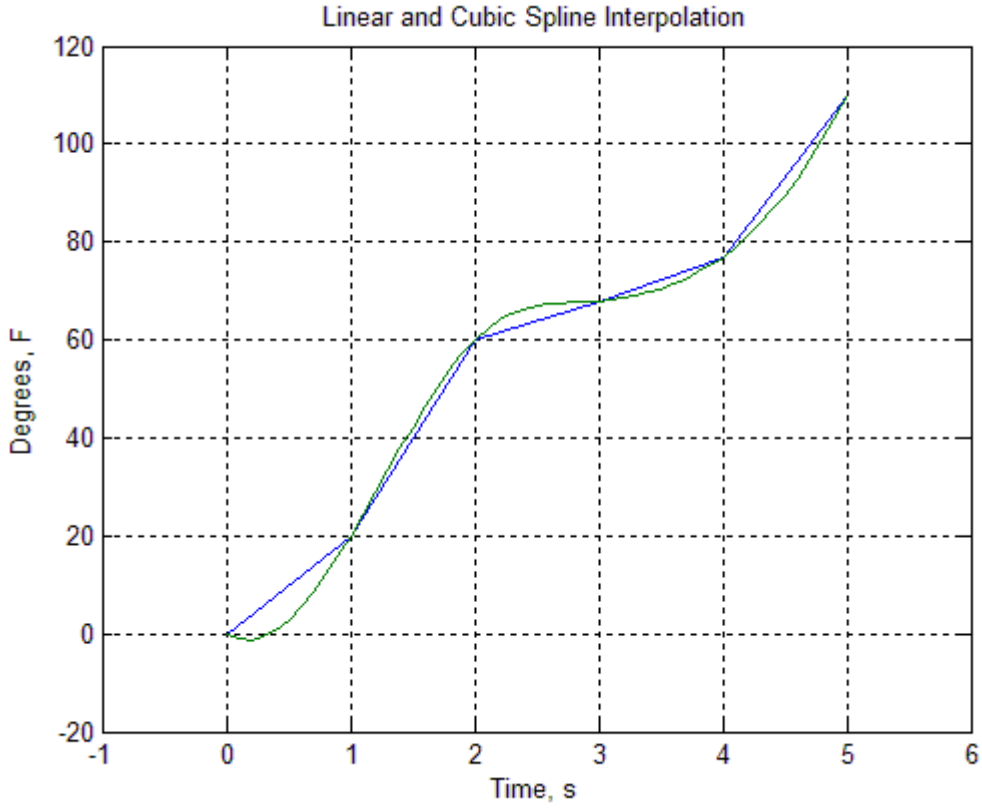
შესავალი

ვთქვათ გვაქვს ექსპერიმენტის ან რაიმე ფიზიკურ პროცესზე დაკვირვების შედეგად მიღებული მონაცემები. ეს მონაცემები სზოგადოდ შეიძლება განვიხილოთ როგორც $y=f(x)$ შესაბამისი წერტილიების კოორდინატები. გვინდა გამოვთვალოთ ამ ფუნქციის მნიშვნელობები x ისეთი მნიშვნელობებისათვის, რომელთა შესახებ ინფორმაცია არ გვაქვს. მაგალითად, გვაქვს მონაცემები (a, f(a)) და (c, f(c)). თუ გვსურს შევაფასოთ ფუნქციის მნიშვნელობა b წერტილში f(b), სადაც a<b<c, უნდა დავუშვათ, რომ f(a) და f(c) შეერთებულია წრფით და წრფივი ინტერპოლირების საშუალებით გამოვითვლით f(b). თუ დავუშვებთ, რომ ეს წერტილები შეერთებულია კუბური პოლინომის საშუალებით, f(b) გამოსათვლელად გამოვიყენებთ კუბური ინტერპოლირების მეთოდს. ზოგჯერ გვჭირდება მონაცემების მიხედვით

დასმულ წერტილებზე მრუდი ისე გავატაროთ, რომ რაც შეიძლება კარგად მოერგოს დასმულ წერტილებს, რამდენადაც ეს შესაძლებელია, ახლოს გაიაროს ყველა წერტილთან. ამ ამოცანას ემსახურება უმცირეს კვადრატთა მეთოდი, რომელიც ისე შეარჩევს მრუდს, რომ მანძილი მოცემულ წერტილსა და ფუნქციით გამოთვლილ შესაბამის მნიშვნელობას შორის უმცირესი იყოს ყველა წერტილისათვის. ახლა განვიხილავთ ინტერპოლაციისა და მრუდის გამოთვლის (curve fitting) მაგალითებს.

4.1 ინტერპოლირება

განვიხილავთ ინტერპოლირების ორ ტიპს: წრფივი და კუბური (spline) ინტერპოლირება. ორივე შემთხვევაში ვუშვებთ, რომ გვაქვს მონაცემები, რომელიც წარმოდგენილია წერტილის კოორდინატების სახით (x, f(x)). გვინდა შევაფასოთ სიდიდე f(b), რომლის მნიშვნელობა არ გვაქვს საწყის მონაცემებში, მაგრამ სრულდება პირობა a<b<c და ვიცით f(a) და f(c) მნიშვნელობები. ნახ. 9.1 ნაჩვენებია 6 წერტილი, რომლებიც შეერთებულია მონაკვეთებით (სწორი ხაზით) და მესამე რიგის – კუბური პოლინომური წირით. რასაკვირველია ფუნქციის შუალედური მნიშვნელობები დამოკიდებულია იმაზე, თუ ინტერპოლირების რომელ მეთოდს ავირჩევთ მათ განსასაზღვრავად.



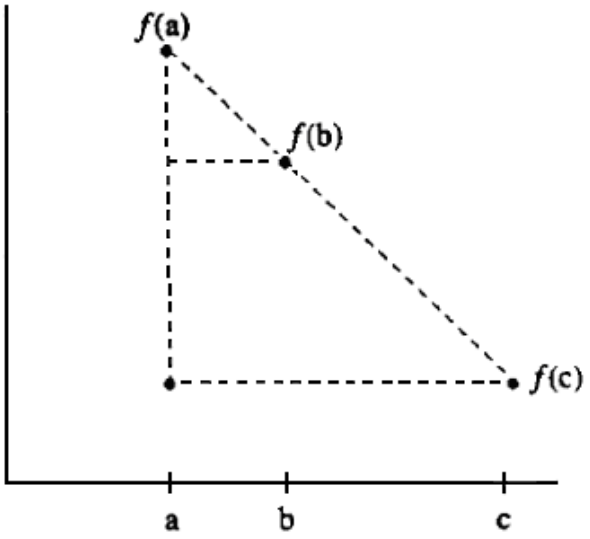
ნახ. 9.1 წრფივი და კუბური ინტერპოლირება

9.1.1 წრფივი ინტერპოლირება

ძირითადი მეთოდი ორ მოცემულ წერტილს შორის მოთავსებული წერტილის კოორდინატის შესაფასებლად წრფივი ინტერპოლირების მეთოდია. ნახ. 9.2–ზე გვაქვს ორი წერტილი. თუ

დავუშვებთ, რომ ფუნქცია ამ ორ წერტილს შორის შეიძლება შეფასდეს როგორც წრფე, რომელიც ამ წერტილებზე გადის, ფუნქციის მნიშვნელობა მათ შორის მოთავსებული ნებისმიერი წერტილისათვის შეგვიძლია გამოვთვალოთ შემდეგი ფორმულით:

$$\frac{f(a)-f(b)}{b-a} = \frac{f(a)-f(c)}{c-a}$$



ნახ. 9.2 მსგავსი სამკუთხედები

$$f(b) = f(a) + \frac{b-a}{c-a}(f(c) - f(a))$$

ვთქვათ ჩვენს ხელთაა ექპერიმენტული მონაცემები. შედარებით მარტივი საქმეა ინტერპოლირების საშუალებით მივიღოთ შუალედური მნიშვნელობები. ინტერპოლირების გამოყენება შესაძლებელია მას შემდეგ, რაც ჩვენს ხელთ არსებულ მონაცემებს შორის ვიპოვით იმ ორ წერტილს, რომელთა შორისაც ჩვენთვის საჭირო წერტილი მდებარეობს. შესაძლებელია ასეთი წერტილები ვერც ვიპოვოთ. MATLAB საშუალებით წრფივი ინტერპოლირება ხორციელდება შემდეგი ფუნქციების საშუალებით. interp1 და interp2.

interp1(x,y,x1)– ფინქცია ახორციელებს ერთგანზომილებიან წრფივ ინტერპოლირებას მონაცემთა ცხრილის საფუძველზე. ვთქვათ გვაქვს მონაცემები x ვექტორის სახით და მისი შესაბამისი მონაცემები y ვექტორის სახით და x1 - ის მნიშვნელობა, რომლისთვისაც გვსურს შესაბამისი y1 სიდიდის ინტერპოლირება. ფუნქცია მოძებნის x ვექტორში იმ მნიშვნელობებს, რომელთა შორისაც x1 სიდიდეა მოთავსებული და წრფივი ინტერპოლირების ფორმულის საშუალებით გამოითვლის y1 სათანადო მნიშვნელობას. შევნიშნავთ, რომ x ვექტორში მონაცემები ზრდის მიხედვით უნდა იყოს დალაგებული, თუ ეს ასე არ არის, MATLAB მიგვითითებს შეცდომაზე. ამასთან ჩვენს მიერ მითითებული სიდიდე უნდა თავსდებოდეს მონაცემთა პირველ და უკანასკნელ მნიშვნელობას შორის, განსხვავებულ შემთხვევაში MATLAB მოგვცემს მნიშვნელობას NAN.

ამ ფუნქციის საილუსტრაციოდ განვიხილოთ ტემპერატურის მონაცემები, რომელიც გაზომილია ცილინდრის ხუფში ავტომობილის ახალი ძრავის გამოცდისას.

დრო, წმ	ტემპერატურა, °F
0.0	0.0

1.0	20.0
2.0	60.0
3.0	68.0
4.0	77.0
5.0	110.0

პირველ რიგში ამ ინფორმაციას ჩავწერთ მატრიცის სახით, სადაც x ვექტორი იქნება დრო, y – ტემპერატურა.

```
x = [0.0, 1.0, 2.0, 3.0, 4.0, 5.0]';
y = [0.0, 20.0, 60.0, 68.0, 77.0, 110.0]';
```

ამის შემდეგ შეგვიძლია ვისარგებლოთ **interp1** ფუნქციით. დროის ისეთი მნიშვნელობებისათვის, რომელიც თავსდება ინტერვალში [0 – 5] წამი. მაგალითად, განვიხილოთ შემდეგი ბრძანებები:

```
y1 = interp1(x,y, 2.6);
y2 = interp1(x,y, 4.9);
```

მივიღებთ y1 = 64.8, y2 = 106.7.

თუ y მატრიცაა და xi ვექტორი, ინტერპოლირება შესრულდება y ყველა სვეტისათვის და მივიღებთ y1 მატრიცას, რომლის სტრიქონების რაოდენობა xi ვექტორის სიგრძის ტოლი იქნება, ხოლო სვეტების რაოდენობა - y ვექტორის სვეტების რაოდენობის. მაგალითად, დავუშვათ ტემპერატურა გავზომეთ ცილინდრის ხუფის სამ სხვადასხვა წერტილში და მივიღეთ შემდეგი ცხრილი:

დრო, წმ	ტემპ. 1, °F	ტემპ. 2, °F	ტემპ. 3, °F
0.0	0.0	0.0	0.0
1.0	20.0	25.0	52.0
2.0	60.0	62.0	90.0
3.0	68.0	67.0	91.0
4.0	77.0	82.0	93.0
5.0	110.0	103.0	96.0

ავაგოთ სამსვეტიანი მატრიცა y ტემპერატურისათვის სამ სხვადასხვა წერტილში:

```
y(: , 1) = [0.0, 20.0, 60.0, 68.0, 77.0, 110.0]';
y(: , 2) = [0.0, 25.0, 62.0, 67.0, 82.0, 103.0]';
y(: , 3) = [0.0, 52.0, 90.0, 91.0, 93.0,  96.0]';
```

თუ გვინდა მივიღოთ ინტერპოლირებული მნიშვნელობა ცილინდრის ხუფის სამივე წერტილში დროის მომენტში 2.6 წამი, MATLAB უნდა მივცეთ ბრძანება:

```
temps = interp1(x,y, 2.6)
```

მივიღებთ:

```
temps =
```

```
    64.8000   65.0000   90.6000
```

interp2 – ახორციელებს წერტილის ინტერპოლაციას ორგანზომილებიან ზედაპირზე z=f(x,y)

9.1.2 კუბური ინტერპოლირება - კუბური სპლაინი

კუბური სპლაინი (წირი) არის გლუვი წირი, რომელიც გაივლის მოცემულ წერტილებზე ისე, რომ წირის მონაკვეთი ყოველ ორ მომდევნო წერტილს შორის მესამე ხარისხის პოლინომს წარმოადგენს. 4.1 ნაჩვენებია კუბური წირი, რომელიც 6 წერტილს აერთებს. იმისათვის, რომ ეს წირი მივიღოთ 5 სხვადასხვა კუბური განტოლება უნდა ამოვხსნათ.

MATLAB-ში კუბური წირი გამოითვლება ფუნქციით **spline**. მისი პირველი ორი არგუმენტი x, y წარმოადგენს ვექტორებს, იმ წერტილების კოორდინატებს, რომლებიც წირმა უნდა შეაერთოს. მესამე არგუმენტი შეიცავს x კოორდინატებს, რომელთა შესაბამისი y უნდა გამოვთვალოთ მესამე რიგის ინტერპოლაციით. x ვექტორის მნიშვნელობები ზრდის მიხედვით უნდა იყოს დალაგებული. **spline** ფუნქციის საილუსტრაციოდ დავუბრუნდეთ ცილინდრის ხუფის ტემპერატურათა მონაცემებს. დროის მნიშვნელობისათვის 2.6 წამი გამოვთვალოთ ტემპერატურა არა წრფივი, არამედ კუბური ინტერპოლირების საშუალებით:

```
x = [0, 1, 2, 3, 4, 5];
y = [0.0, 20.0, 60.0, 68.0, 77.0, 110.0];
temp1 = spline(x,y,2.6)
```

მივიღებთ:

```
temp1 =

   67.3013
```

თუ გვსურს გამოვიყენოთ კუბური ინტერპოლირება ტემპერატურის გამოსათვლელად დროის ორ სხვადასხვა მომენტში, MATLAB ბრძანება ასეთი იქნება:

```
temp2 = spline(x,y,[2.6,4.9])
```

მივიღებთ:

```
temp2 =

    0.6730  1.0520
```

თუ გვინდა მივიღოთ კუბური წირი x მნიშვნელობათა ინტერვალში მთლიანად, უნდა შევქმნათ x ვექტორი საჭირო გარჩევით და ავიღოთ იგი როგორც მესამე პარამეტრი **spline** ფუნქციისათვის.

მაგალითად, ნახ. 9.1 ნაჩვენები კუბური წირი მიიღება შემდეგი ბრძანებების შედეგად:

```
x=[0.0, 1.0, 2.0,3.0,4.0,5.0];
y=[0, 20, 60, 68, 77, 110];
newx=0:0.1:5;
newy=spline(x,y,newx);
plot(x,y,newx,newy),...
axis([-1 6 -20 120]),...
title('Linear and Cubic Spline Interpolation'),...
xlabel('Time, s'),...
ylabel('Degrees, F'),...
grid
```

სწორხაზოვანი მონაკვეთები შეესაბამება ბრძანებას **plot**(x,y) და წარმოადგენს წრფივ ინტერპოლირებას, ხოლო გლუვი წირი - **plot**(newx,newy) ბრძანების შედეგია და კუბური ინტერპოლირების შედეგს წარმოადგენს.

სავარჯიშო

დავუშვათ გვაქვს მონაცემთა შემდეგი მწკრივი:

დრო [წმ]	ტემპერატურა, [°F]
0.0	72.5
0.5	78.1
1.0	86.4
1.5	92.3
2.0	110.6
2.5	11.5
3.0	109.3
3.5	110.2
4.0	110.5
4.5	109.9
5.0	110.2

1. შეაერთეთ წერტილები წრფის მონაკვეთებით და კუბური წირით
2. MATLAB საშუალებით გამოითვალეთ ტემპერატურა წრფივი ინტერპოლირებით დროის შემდეგი მნიშვნელობებისათვის

   0.3, 1.25, 2.36, 4.48
3. MATLAB საშუალებით გამოითვალეთ ტემპერატურა კუბური ინტერპოლირებით დროის შემდეგი მნიშვნელობებისათვის

   0.3, 1.25, 2.36, 4.48
4. გამოიყენეთ წრფივი ინტერპოლირების მეთოდი MATLAB –ში და გამოითვალეთ დროის მნიშვნელობები, რომელიც შეესაბამება ტემპერატურათა შემდეგ მნიშვნელობებს:

   81, 96, 100, 106
5. გამოიყენეთ კუბური ინტერპოლირების მეთოდი MATLAB –ში და გამოითვალეთ დროის მნიშვნელობები, რომელიც შეესაბამება ტემპერატურათა შემდეგ მნიშვნელობებს:

   81, 96, 100, 106

პრობლემა: რობოტის მკლავის მანიპულატორი

იმისათვის, რომ რობოტმა რაიმე მოქმედება შეასრულოს, მას უნდა გააჩნდეს კონტროლის სისტემით მართვადი მანიპულატორები, რომელიც შეიმუშავებს მოძრაობის მარშრუტს, რათა იგი გადაადგილდეს ერთი მდებარეობიდან მეორეზე. მარშრუტი გლუვი უნდა იყოს, რათა თავიდან იქნას აცილებული მკვეთრი მოძრაობები, რომლის დროსაც შესაძლოა დაზიანდეს ობიექტი ან თვითონ რობოტის მკლავი. ამიტომ მარშრუტი რობოტის მკლავისათვის განისაზღვრება წერტილების ერთობლიობით, რომელთა გასწვრივაც მკლავმა უნდა იმოძრაოს. იმისათვის, რომ გადაადგილება გლუვი მრუდის გასწვრივ მოხდეს, ინტერპოლირებას მიმართავენ. განვიხილავთ მსგავს პრობლემას, იმ დაშვებით, რომ რობოტის მანიპულატორი მკლავი მოძრაობს სიბრტყეზე, თუმცა საზოგადოდ მოძრაობა სივრცეში ხორციელდება.

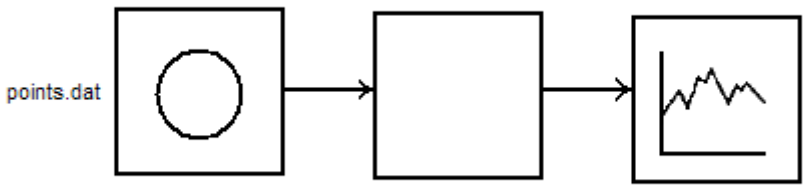
ალგორითმის შემუშავების, ან პრობლემის ამოხსნის მნიშვნელოვანი პირობაა ყურადღებით განვიხილოთ ხომ არ არის რაიმე განსაკუთრებული შემთხვევა, რომელიც მხედველობიდან არ უნდა გამოგვრჩეს. ამ ამოცანაში ვუშვებთ, რომ იმ წერტილების კოორდინატები, რომელთა გასწვრივ რობოტის მკლავმა უნდა იმოძრაოს, მოთავსებულია მონაცემთა ფაილში და ისეთი რიგით არის დალაგებული, რომ მკლავმა უნდა მიაღწიოს გარკვეულ მდებარეობას, აიღოს იქ მდებარე ობიექტი, შემდეგ გადაადგილდეს წერტილში, სადაც დატოვებს აღებულ ობიექტს და კვლავ დაუბრუნდეს საწყის მდებარეობას. ასევე ვუშვებთ, რომ მონაცემებში ჩართულია შუალედური წერტილები, რომელიც საშუალებას აძლევს მას თავი აარიდოს შემხვედრ დაბრკოლებას და ხელი არ შეუშალოს სენსორებს, რომლებიც მონაცემებს აგროვებენ. ყოველ წერტილს გააჩნია სამი კოორდინატი: x და y საწყისი მდებარეიბის მიმართ და მესამე, რომელიც კოდირებულია შემდეგნაირად:

კოდი	ინტერპრეტაცია
0	საწყისი მდებარეობა
1	შუალედური მდებარეობა
2	ობიექტის აღების მდებარეობა
3	ობიექტის დადების მდებარეობა

ამოცანის დასმა

უნდა გავატაროთ კუბური მრუდი მოცემულ წერტილებზე, რომელიც უნდა გაიაროს რობოტის მკლავის მანიპულატორმა თავის გზაზე, საწყისი მდებარეობიდან ობიქტის აღების, მისი დაბინავების გავლით კვლავ საწყის მდებარეობამდე.

1. INPUT/OUTPUT დიაგრამა



ნახ. 9.3 I/O დიაგრამა

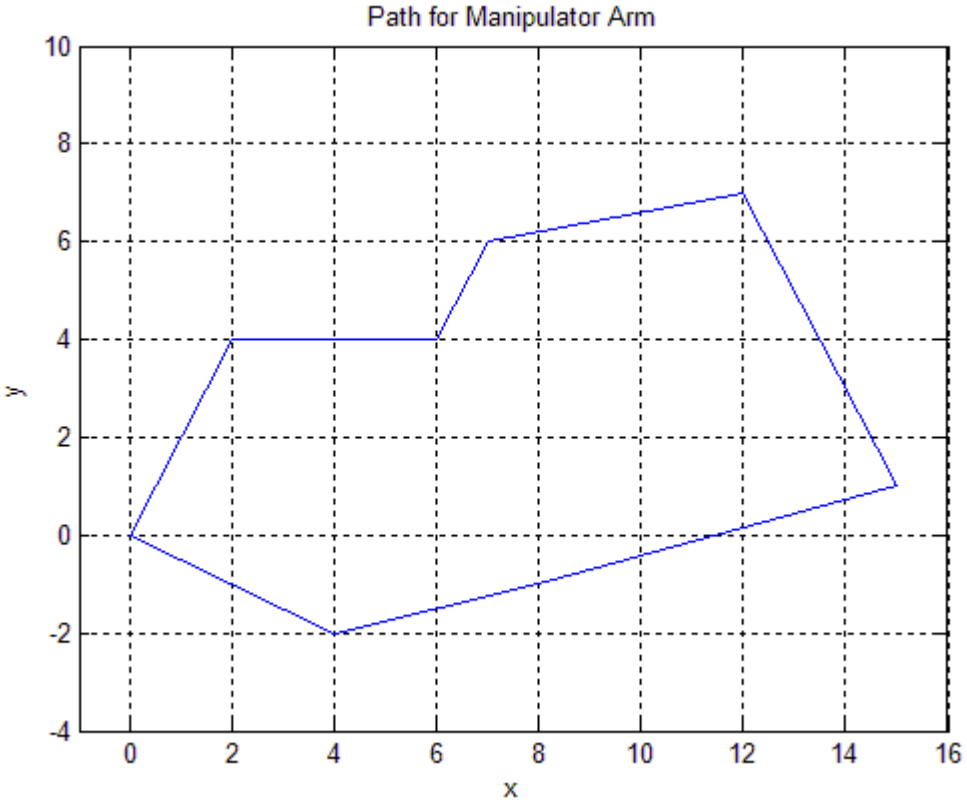
2. სახელდახელო ამოხსნა

განვსაზღვროთ, ხომ არ არის რაიმე განსაკუთრებული შემთხვევა, რომელიც მხედველობაში უნდა მივიღოთ ალგორითმის შედგენისას. განვიხილოთ მონაცემთა ფაილი, რომელიც შეიცავს წერტილებს შემდეგი მონაცემებით:

x	y	კოდი	კოდის ინტერპრეტაცია
0	0	0	საწყისი მდებარეობა
2	4	1	შუალედური მდებარეობა
6	4	1	შუალედური მდებარეობა
7	6	2	ობიექტის აღების მდებარეობა
12	7	1	შუალედური მდებარეობა
15	1	3	ობიექტის დაბინავების მდებარეობა

8	-1	1	შუალედური მდებარეობა
4	-2	1	შუალედური მდებარეობა
0	0	0	საწყისი მდებარეობა

სწორი ხაზებით შეერთებული ეს წერტილები ნაჩვენებია ნახ. 9.4 – ზე.
დავყოთ მარშრუტი სამ ნაწილად: საწყისი მდებარეობიდან ობიექტის აღების მდებარეობამდე, ობიექტის აღების მდებარეობიდან მისი დაბინავების მდებარეობამდე და აქედან უკან საწყის მდებარეობამდე. ასე ორი მიზეზის გამო მოვიქეცით: პირველი, მანიპულატორი უნდა შეჩერდეს ყოველი მათგანის დსრულებისას, ასე, რომ ეს მონაკვეთები რეალურად გამოყოფილი მონაკვეთებია. მეორე – ფუნქცია **spline** მოითხოვს, რომ x კოორდინატები ზრდის მიხედვით იყოს დალაგებული. ამგვარად გზის ყოველ მონაკვეთში კოორდინატები ზრდის მიხედვით უნდა იყოს დალაგებული. დავუშვებთ, რომ ეს მომენტი გათვალიწინებულია მონაცემების შედგენისას



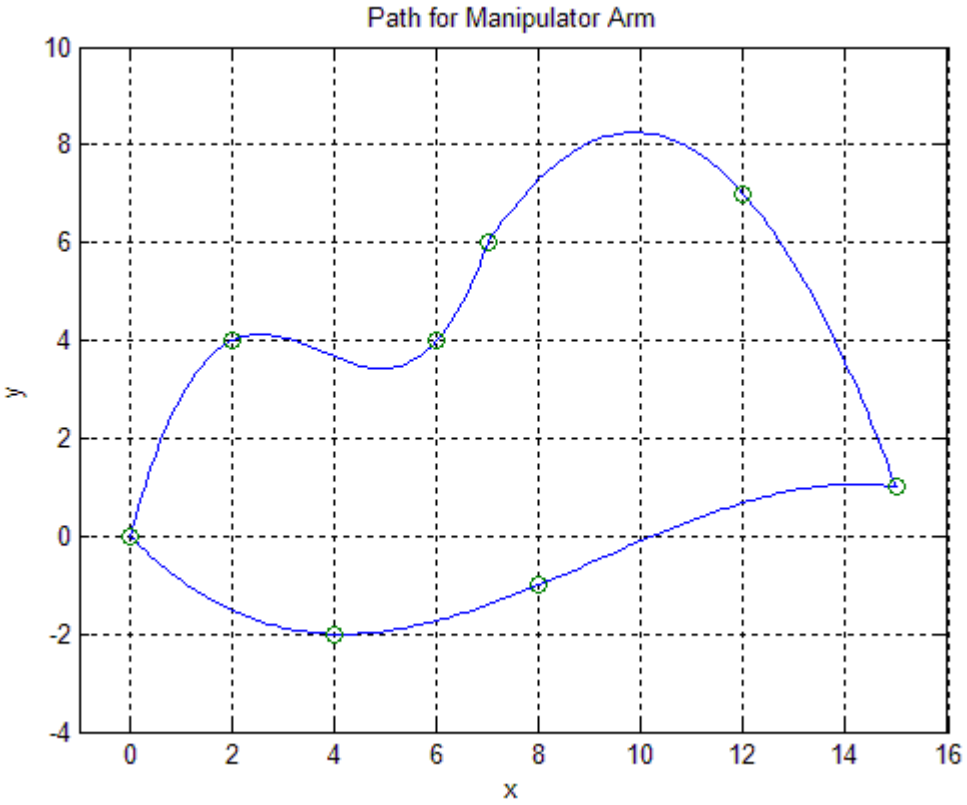
ნახ. 9.4 სწორი ხაზებით შეერთებული წერტილები მონაცემთა ფაილიდან რობოტის მანიპულატორისათვის

3. MATLAB ამოხსნა

მინაცემთა სამ ნაწილად გაყოფა მნიშვნელოვანი საფეხურია ალგორითმის შედგენაში. ნაკლებად მნიშვნელოვანია ის თუ რამდენ წერტილს შევარჩევთ კუბური წირისათვის. რადგან კოორდინატი შეიძლება იყოს ძალიან მცირე და ძალიან დიდი სიდიდე, გადავწყვიტეთ

ვიპოვოთ უმცირესი მანძილი წერტილებს შორის და მისი მეათედი გამოვიყენოთ როგორც ნაზრდი ინტერპოლირებისათვის საჭირო წერტილების x კოორდინატების შერჩევისათვის. ამ მოსაზრებით ყოველ ორ მოცემულ წერტილს შორის მინიმუმ 10 ინტერპოლირებული წერტილი მაინც გვექნება.

```
%     This program reads the data file contaning the
%     points for a path for a manipulator arm to go to
%     a location to grasp an object, then move to
%     another location to release the object, and
%     then move back to the start posision

load points.dat;
x=points(:,1);
y=points(:,2);
code=points(:,3);
%
%     Generate the three separate paths
%
grasp=find(code==2);
release=find(code==3);
lenx=length(x);
x1=x(1:grasp);           y1=y(1:grasp);
x2=x(grasp:release);     y2=y(grasp:release);
x3=x(release:lenx);      y3=y(release:lenx);
%
%     Compute time sequences
%
incr=min(abs(x(2:lenx)-x(1:lenx-1)))/10;
t1=x(1):incr*sign(x(grasp)-x(1)):x(grasp);
t2=x(grasp):incr*sign(x(release)-x(grasp)):x(release);
t3=x(release):incr*sign(x(lenx)-x(release)):x(lenx);
%
%     Compute spline path
%
s1=spline(x1,y1,t1);
s2=spline(x2,y2,t2);
s3=spline(x3,y3,t3);
%
%     Plot spline path
%
axis([-1, 16, -4, 10])
plot([t1 t2 t3],[s1 s2 s3], …
[x1' x2' x3'],[y1' y2' y3'],'o'),...
title('Path for Manipulator Arm'),...
xlabel('x'), ylabel('y'), grid
```



ნახ. 9.5 კუბური ინტერპოლირების შედეგად მიღებული წირით შეერთებული წერტილები

4. შემოწმება

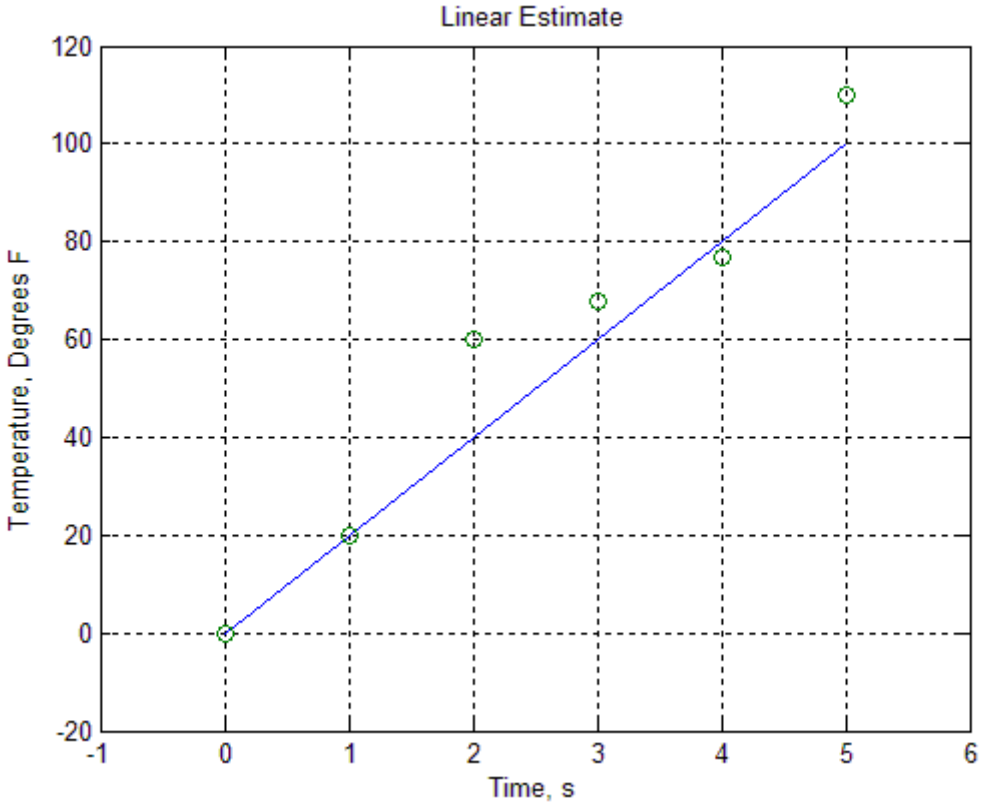
შევამოწმოთ პროგრამა მონაცემთა ზემოთგანხილული ფაილის მაგალითზე. მივიღებთ ნახ. 9.5.

4.2 უმცირეს კვადრატთა მეთოდი

დავუშვათ გვაქვს ექსპერიმენტული მონაცემები, ავაგეთ გრაფიკი და ცხადად ჩანს, რომ ისინი ძირითადად წრფის გასწვერივ დალაგდნენ. თუმცა თუ გავავლებთ წრფეს, მხოლოდ რამდენიმე მათგანი გადაიკვეთება. უმცირეს კვადრატთა მეთოდი გამოიყენება ისეთი წრფის მოსაძებნად, რომელიც ყველა წერტილს ყველაზე ახლოს ჩაუვლის წერტილიდან წრფემდე მანძილის კვადრატის მინიმალურის მოძებნის საფუძველზე. თუმცა ეს წრფე განიხილება, როგორც ოპტიმალური 'მიახლოება' (best fitting), შეიძლება ისეც მოხდეს, რომ არცერთი წერტილი არ მოხვდეს უშუალოდ მასზე. (უნდა აღინიშნოს, რომ ეს მეთოდი ბევრად განსხვავდება ინტერპოლირებისაგან, რადგან წირი, რომლითაც ვსარგებლობთ წრფივი და კუბური ინტერპოლირებისას, გაივლის ყველა წერტილზე.) ამ თავში განვიხილავთ ჯერ წრფის, შემდეგ კი პოლინომური წირის გავლებას ექსპერიმენტული მონაცემების შესაბამის წერტილებზე უმცირე კვადრატთა მეთოდის გამოყენებით.

9.1.3 წრფივი რეგრესია

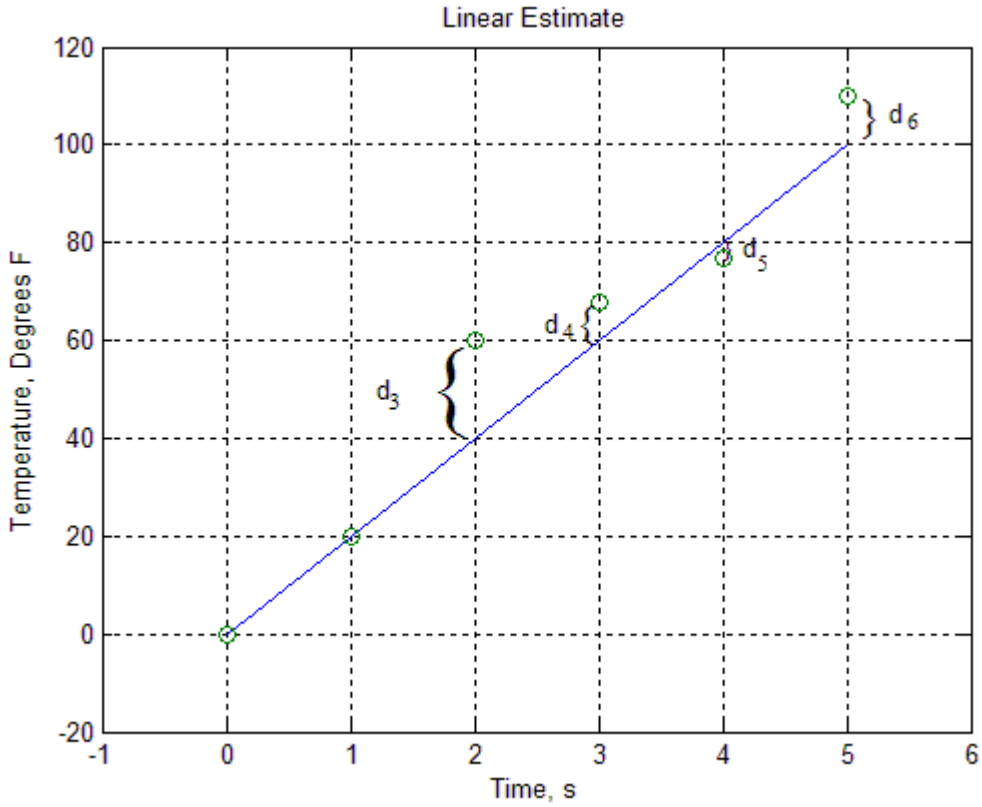
წრფივი რეგრესია ისეთ პროცესს ეწოდება, რომელიც განსაზღვრავს ისეთ წრფივ განტოლებას, რომელიც ოპტიმალურად გაივლის არსებული მონაცემების შესაბამის წერტილებზე, წრფემდე მანძილების კვადრატების ჯამის მინიმიზაციის საფუძველზე. კარგად რომ გავიგოთ ეს პროცესი დავუბრუნდეთ ტემპერატურის მონაცემებს, რომლის ანათვალიც აღებულია ახალი ძრავის ცილინდრის ხუფში. თუ ავაგებთ ამ მონაცემებს, ვნახავთ, რომ ისინი წრფის გასწვრივ განლაგდებიან. წრფე y=20x ერთი შეხედვით კარგად ასახავს წერტილთა მდებარეობას. ნახ. 9.6 ნაჩვენებია ეს წერტილები და გავლებული წრფე. შემდეგი ბრძანებები იძლევა ნახ. 9.7.



ნახ. 9.6 წრფივი შეფასება

```
x = [0, 1, 2, 3, 4, 5 ];
y = [0, 20, 60, 68, 77, 110 ];
y1 = 20*x;
plot(x,y1,x,y,'o'), title('Linear Estimate'),...
xlabel('Time, s'), ylabel('Temperature, Degrees F'), grid
axis([-1,6,-20,120])
```

ვნახოთ რამდენად კარგად ასახავს ეს წრფივი შეფასება ჩვენს მონაცემებს. პირველ რიგში გავზომოთ მანძილები წერტილებიდან წრფემდე. ეს მანძილები ნაჩვენებია ნახ. 9.7-ზე



ნახ. 9.7 მანძილი წერტილსა და წრფივ შეფასებას შორის

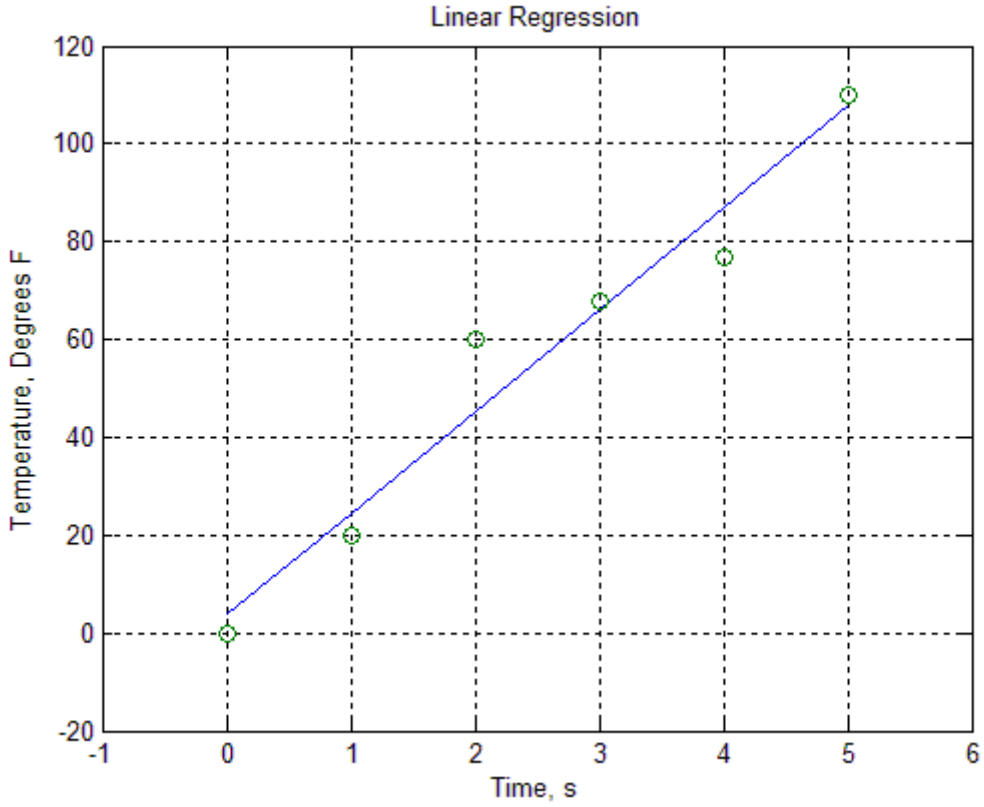
პირველი ორი წერტილი პირდაპირ წრფეზე მოთავსდა, ასე, რომ d_1 და d_2 0-ის ტოლია. d_3 ტოლია 60-40, ანუ 20. ასევე შეიძლება შევაფასოთ დანარჩენი მანძილებიც. თუ გამოვითვლით მანძილების ჯამს, დადებითმა და უარყოფითმა მანძილებმა შეიძლება ერთმანეთი გააბათილონ და მივიღებთ უფერო მცირე მნიშვნელობას, ვიდრე მოსალოდნელია, ამის თავიდან ასაცილებლად უნდა შევკრიბოთ აბსოლუტური სიდიდეები ან კვადრატში აყვანილი მნიშვნელობები. წრფივი შეფასების ხარისხის ზომად ვირჩევთ წერტილებსა და წრფივ შეფასებას შორის მანძილების კვადრატების ჯამს. ეს ჯამი გამოითვლება MATLAB საშუალებით შემდეგი ბრძანებებით:

```
sum_sq = sum((y-y1).^2)
```

მივიღებთ: sum_sq =573.

თუ გავავლებთ სხვა წრფეს, შეგვიძლია ასეთივე ჯამი მისთვისაც გამოვითვალოთ. ამ ორ წრფეს შორის უკეთესი შეფასება ის იქნება, რომლისთვისაც ჯამის მნიშვნელობა უფრო მცირეა. იმისათვის, რომ ვიპოვოთ საუკეთესო წრფივი შეფასება, რომლისთვისაც კვადრატული ჯამის მნიშვნელობა უმცირესია, უნდა დავწეროთ განტოლება, რომელიც გამოითვლის კვადრატული ჯამის მნიშვნელობებს წრფის ზოგად ფორმულაზე დაყრდნობით: y = mx + b. დავწერთ განტოლებას, რომელიც წარმოადგენს მანძილების კვადრატების ჯამს. ამ განტოლების ცვლადებია m და b. გამოვითვლით ამ განტოლების წარმოებულს m-ით და b –ით და გავუტოლებთ მას 0. ამგვარად გამოთვლილი m და b განსაზღვრავს იმ წრფეს, რომელსაც ექნება მანძილების კვადრატების ჯამის უმცირესი მნიშვნელობა მოცემული წერტილების მიმართ. ეს სიდიდები შეიძლება გამოთვლილი იქნას ფუნქციით **polyfit**. ეს ფუნქცია წარმოდგენილი იქნება მოგვიანებით, პოლინომური რეგრესიის განხილვის შემდეგ. ნახ. 9.8 წარმოადგენს წრფეს უმცირესი კვადრატული გადახრით, რომელიც ოპტიმალურად

ასახავს ჩვენს მონაცემებს. იგი შეესაბამება ოპტიმალურ წრფივ შეფასებას. გადახრის კვადრატების ჯამი ტოლია 356.82



ნახ. 9.8 წრფივი რეგრესია უმცირესი კვადრატული გადახრით

9.1.4 პოლინომური რეგრესია

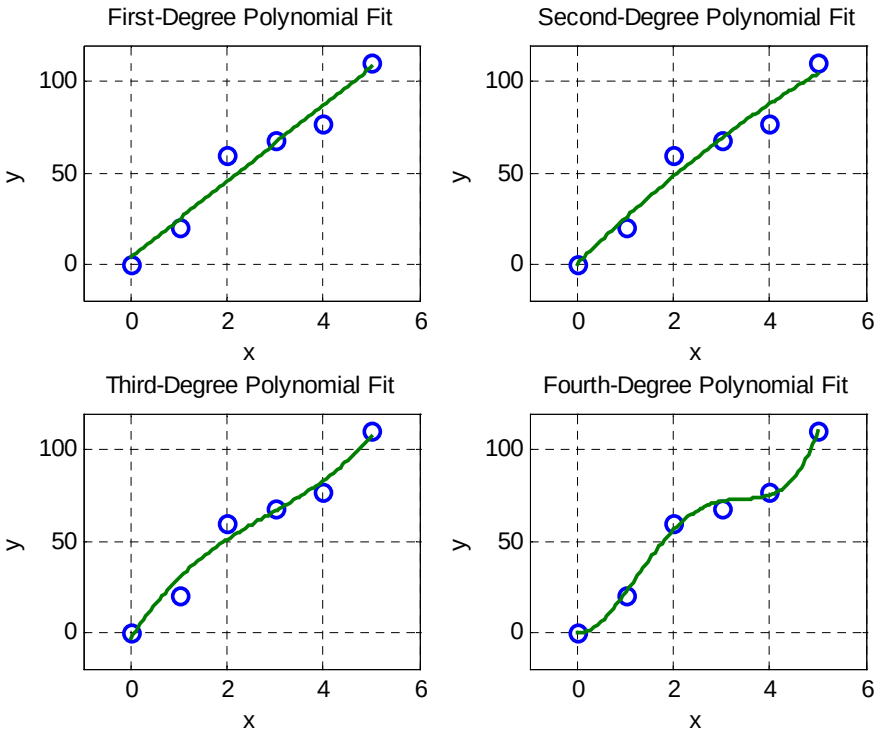
უკვე წარმოგიდგინეთ უმცირეს კვადრატთა მეთოდი, რომელიც გამოითვლის ექსპერიმენტული მონაცემების ოპტიმალურად ამსახველ წრფეს. იგივე შედეგს მივაღწევთ პოლინომური რეგრესიის მეთოდის გამოყენებით, წრფე ხომ იგივე პირველი რიგის პოლინომია. შეგახსენებთ, რომ ერთცვლადიანი პოლინომის ზოგადი ფორმულაა:

$$f(x) = a_0 x^N + a_1 x^{N-1} + a_2 x^{N-2} + \cdots + a_{N-1} x + a_N$$

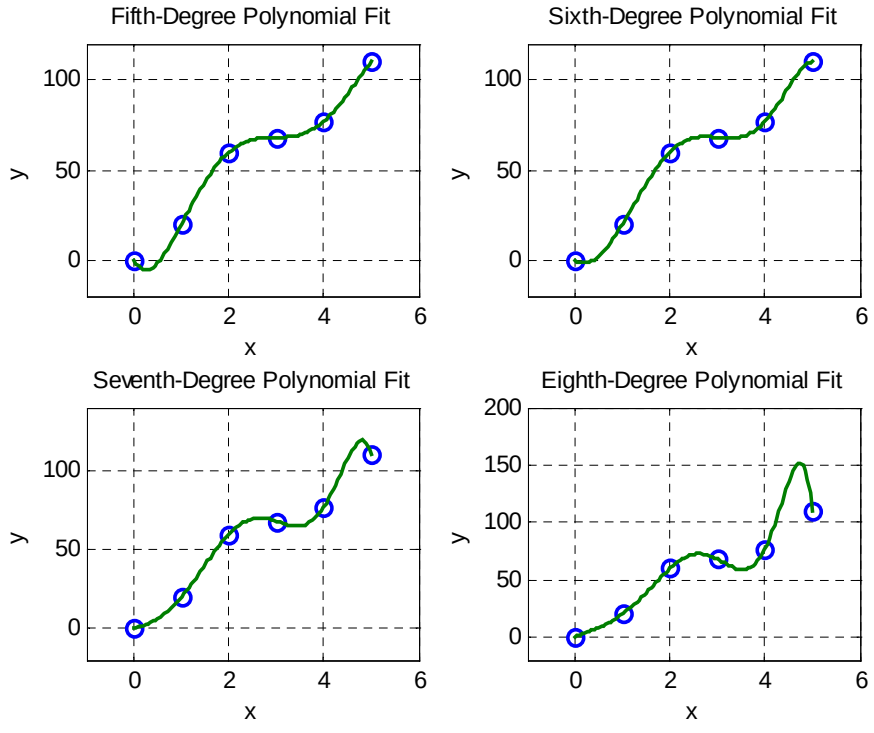
პოლინომის ხარისხი მისი ცვლადის ხარისხის უმაღლესი მაჩვენებლის ტოლია. ამრიგად კუბური, მესამე ხარისხის პოლინომის ზოგადი ფორმულა ასეთია:

$$g(x) = a_0 x^3 + a_1 x^2 + a_2 x + a_3$$

ნახ. 9.9 და ნახ. 9.10 ნაჩვენებია ახალი ძრავის ცილინდრის ხუფში ათვლილი ტემპერეტურის მონაცემების სხვადასხვა რიგის პოლინომური რეგრესიის შედეგად მიღებული მრუდები.



ნახ. 9.9 პოლინომური რეგრესიის შედეგად მიღებული მრუდები



ნახ. 9.10 პოლინომური რეგრესიის შედეგად მიღებული მრუდები

ყურადღება მიაქციეთ, რაც უფრო იზრდება პოლინომის რიგი, მით უფრო მეტია წერტილების რაოდენობა, რომელზეც წირი უშუალოდ გაივლის. თუ მოცემულო გვაქვს N წერტილი და გამოვიყენებთ N რიგის პოლინომურ რეგრასიას, მიღებული წირი ყველა წერტილზე გაივლის.

მაღალი რიგის პოლინომური შეფასებისას ყურადღება უნდა მივაქციოთ ჩვენს ხელთ არსებულ მონაცემთა თავისებურებებს. მაღალი რიგის პოლინომს წერტილებს შორის შესაძლოა ცვალებადობის ფართო ინტერვალი ახასიათებდეს, რამაც შეიძლება მცდარ გზაზე დაგვაყენოს მონაცემების შეფაებისას გარკვეულ ინტერვალში.

9.1.5 polyfit და polyval ფუნქცია

MATLAB –ში პოლინომური რეგრასია ხორციელდება ფუნქციით **polyfi**t. ფუნქციას აქვს სამი არგუმენტი. მონაცემთა x - y კოორდინატები და პოლინომის რიგი N. ფუნქცია გვაძლევს N რიგის პოლინომის კოეფიციენტებს პოლინომის x ცვლადის ხარისხის კლების მიმართულებით. (ყურადღება მიაქციეთ, N რიგის პოლინომისათვის ვღებულობთ N+1 კოეფიციენტს.)

ნახ. 9.8 გამოსახული მონაცემების საუკეთესო წრფივი შეფასების გადხრის კვადრატების ჯამი = 356.82. მისი გამოთვლა და მრუდის აგება შესაძლებელიია შემდეგი ბრძანებებით:

```
x = [0, 1, 2, 3, 4, 5];
y = [0 20 60 68 77 110];
coef = polyfit(x,y,1);
m=coef(1);
b=coef(2);
ybest=m*x+b;
sum_sq = sum((y-ybest).^2);
plot(x,ybest,x,y,'o'),...
xlabel('Time, s'), ylabel('Temperature, Degrees F'),...
title('Linear Regression'),...
grid
axis([-1,6,-20, 120])
```

ფუნქცია **polyval** გამითვლის პოლინომს, რომელსაც ახასიათებს უმცირესი კვადრატული გადახრა მოცემული წერტილებიდან. მისი პირველი არგუმენტია პოლინომის კოეფიციენტები, მეორე – x ვექტორი, რომლის არგუმენტებია x ცვლადის ის მნიშვნელობები, რომელთათვისაც გვინდა გამოვითვალოთ პოლინომის მნიშვნელობები. წინა მაგალითში წრფივი რეგრესიის წერტილები გამივითვალეთ მიღებული კოეფიციენტების საშუალებით – ybest=m*x+b, იგივე შედეგს მოგვცემს ფუნქცია **polyva**l:

```
ybest = polyval(coef,x);
```

ავიღოთ მონაცემები, რომლითაც ვსარგებლობდით ზემოხსენებულ მაგალითში და გამოვიყენოთ პოლინომური რეგრესიის მეთოდი, პოლინომის რიგი შევცვალოთ 2-დან 9-მდე. როგორც უკვე აღვნიშნეთ, მოსალოდნელია, რომ დაბალი რიგის პოლინომის შემთხვევაში წირი ყველა მოცემულ წერტილზე არ გაივლის, მაგრამ მეექვსე რიგის პოლინომი უკვე ყველა წერტილს უნდა მოიცავდეს. როგორც ნახზიდან კარგად ჩანს, მეექვსედან მეცხრე რიგის ჩათვლით

ყველა პოლინომი ჩვენს მიერ განხილულ ყველა წერტილს მოიცავს და მათი შესაბამისი უმცირესი კვადრატული ჯამი 0-ის ტოლია, თუმცა დაბალი რიგის პოლინომის შესაბამისი მრუდები უკეთ ასახავს ჩვენი მონაცემების საერთო ტენდენციას. ზემითაღწერილი პოლინომური მრუდები აიგება შემდეგი ბრძანებებით:

```
x =  [ 0, 1, 2, 3, 4, 5 ];
y = [ 0, 20, 60, 68, 77, 110 ];
newx = 0:0.5:5;
for n=1:9;
     f (:,n) = polyval(polyfit(x,y,n), newx) ';
end
```

გრაფიკის ასაგებად ამ ბრძანებებს დაემატება:

```
plot(newx, f(:,2), x,y, 'o')
```

საჭიროა კიდევ დამატებითი ბრძანებები გრაფიკის გასაფორმებლად: სათაურის დაწერა, შესაბამისი წარწერა ღერძებზე, ღერძების საზღვრების განსაზღვრა და subplot ბრძანების გამოყენება.

ამ თავში გავეცანით განსხვავებას ინტერპოლირებასა და უმცირეს კვადრატთა მეთოდს შორის. წარმოგიდგინეთ ინტერპოლირების ორი სხვადასხვა ტიპი: წრფივი და კუბური ინტერპოლირება. განვიხილეთ MATLAB ბრძანებები ინტერპოლირებისათვის. შემდეგ გაგაცანით უმცირეს კვადრატთა მეთოდის ძირითადი პრინციპი და სხვადასხვა რიგის პოლინომური რეგრესია. გიჩვენეთ, თუ როგორ შევარჩიოთ მონაცემებისათვის საუკეთესო წირი უმცირეს კვადრატთა მეთოდის საშუალებით სხვადასხვა რიგის პოლინომის გამოყენებით.

ბრძანებები და ფუნქციები

polyfit	გამითვლის პოლინომს უნცირეს კვადრატთა მეთოდით
polyval	გამოითვლის პოლინომის მნიშვნელობებს
spline	კუბური ინტერპოლირება
interp1	ერთგანზომილებიანი წრფივი ინტერპოლირება
interp2	ორგანზომილებიანი წრფივი ინტერპოლირება

ამოცანები

1 – 7 პრობლემა დაკავშირებულია ამ თავში განხილულ ამოცანებთან, ხოლო 8 – 11 – ახალ საინჟინრო ამოცანებს უკავშირდება

რობოტის მკლავის მანიპულატორი. პრობლემები უკავშირდება ამ თავში განხილულ ამოცანას რობოტის მკლავის მანიპულატორის შესახებ. მონაცემები ჩაწერილია ფაილში points.dat.

1. დაწერეთ პროგრამა იმისათვის, რომ წინასწარ შეამოწმოთ მონაცემები ფაილში points.dat. არის თუ არა დალაგებული ზრდის მოხედვით x კოორდინატის მნიშვნელობები მარშრუტის სამ სხვადასხვა მონაკვეთზე?

2. დავუშვათ ფაილი points.dat მიოცავს სამზე მეტ მონაკვეთს, მაგალითად ეს შეძლება იყოს რამდენიმე ობიექტის გადატანა ახალ ადგილზე. დაწერეთ პროგრამა, რომელიც დაითვლის ინდივიდუალური მარშრუტების რაოდენობას, რომელიც მთავრდება ან ობიექტის აღებით, ან მისი დადებით ახალ ადგილზე, ან საწყის მდებარეობაში დაბრუნებით.
3. შეცვალე კუბური ინტერპოლაციის პროგრამა ისე, რომ მან დაბეჭდოს მთელი მარშრუტის ინტერპოლირებული მონაცემები და ჩაწეროს იგი ფაილში path.dat. წაშალეთ მონაცემები, რომლებიც მეორდება.
4. დაწერეთ პროგრამა, რომელიც წაიკითხავს ფაილს path.dat, რომელიც მე-3 ამოცანაშია აღწერილი, ააგებს საწყის მარშრუტს და დასვამს წრეებს იმ წერტილებზე, სადაც რობოტის მკლავი შეჩერდა. (რობოტის მკლავი ჩერდება რომ აიღოს ობიექტი, დადოს იგი ან იმ შემთხვევაში თუ დაუბრუნდა საწყის მდებარეობას.)
5. შეცვალე მე-4ამოცანაში აღწერილი პროგრამა ისე, რომ მარშრუტი შეიცავდეს სამზე მეტ მონაკვეთს, როგორც მე-2 ამოცანაშია აღწერილი.

ნავთობის ჭაბურღილი.(oil well production) დავუშვათ გვინდა გავიგოთ ნავთობის მოქმედი ჭაბურღილის პროდუქცია როგორაა დამოკიდებული ტემპერატურაზე. გვაქვს ექსპერიმენტული მონაცემები, რომელიც გვიჩვენებს საშუალოდ დღის განმავლობაში წარმოებული ნავთობის რაოდენობას ბარელებში და შესაბამის საშუალოდ ტემპერატურას დღის განმავლობაში. მონაცემები ჩაწერილია ASCII ფაილში oil.dat.

6. რადგანაც მონაცემები არცერთი პარამეტრის მიხედვით არ არის დალაგებული, პირველ რიგში საჭიროა მათი გადალაგება. დაწერეთ პროგრამა, რომელიც კითხულობს ფაილს oil.dat და ქმნის ორ ახალ ფაილს: ფაილი oiltmp.dat უნდა შეიცავდეს მონაცემებს დალაგებულს ნავთობის რაოდენობის ზრდის მიხედვით ტემპერატურას შესაბამისი მონაცემებით, ხოლო ფაილი tmpoil.dat – ტემპერატურის ზრდის მიხედვით დალაგებულ მონაცემებს ნავთობის რაოდენობის შესაბამისი მნიშვნელობებითურთ. თუ გვაქვს წერტილები x კოორდინატის ერთნაირი მნიშვნელობებით, ახალ ფაილში უნდა დარჩეს მხოლოდ ერთი მინაცემი, ხოლო შესაბამისი y უნდა იყოს ერთნაირი x კოორდინატის მქონე y კოორდინატების საშუალო მნიშვნელობა. .
7. დაწერეთ პროგრამა, რომელიც ააგებს tempoil.dat ფაილის მონაცემების გრაფიკს წერტილების მეორე და მესამე რიგის პოლინომური აპროქსიმაციის საფუძველზე. დაბეჭდეთ მიღებული პოლინომის გამოსახულება და უმცირესი კვადრატული ცთომილება.
8. დაწერეთ პროგრამა, რომელიც ააგებს oiltmp.dat ფაილის მონაცემების გრაფიკს წერტილების მეორე და მესამე რიგის პოლინომური აპროქსიმაციის საფუძველზე. დაბეჭდეთ მიღებული პოლინომის გამოსახულება და უმცირესი კვადრატული ცთომილება.
9. დავუშვათ ტემპერატურის ზრდის მიხედვით დალაგებული მონაცემები ნავთობის შესაბამისი რაოდენობით გვინდა შევაფასოთ (აღვწეროთ) მესამე რიგის პოლინომით. დაწერეთ პროგრამა, რომელიც საშუალებას მოგვცემს შევიყვანოთ ტემპერატურის მნიშვნელობა და მივიღოთ ნავთობის რაოდენობის შესაბამისი მნიშვნელობა.

მეტეოროლოგიური ზონდი

10 პოლინომური ანალიზი

პრობლემა: ამინდის წინასწარმეტყველება

ამინდის გამოსაკვლევი ზონდი (მეტეოროლოგიური ბალონი) გამოიყენება ატმოსფეროს ზედა ფენებში მონაცემთა შესაგროვებლად ამინდის მოდელირებისათვის. ბალონი შევსებულია ჰელიუმით და აღწევს წონასწორობის წერტილს როცა სხვაობა ბალონში ჰელიუმისა და გარემოს სიმკვრივეს შორის ნულს უტოლდება. ბალონი ჩერდება მოცემულ სიმაღლეზე. დღის განმავლობაში მზე ათბობს ბალონს, ჰელიუმი ფართოვდება და ბალონში სიმკვრივე ეცემა. ამის გამო ზონდი გადაინაცვლებს ატმოსფეროს უფრო ზედა ფენებში. საღამოს ჰელიუმი კვლავ ცივდება და ზონდი ატმოსფეროს ქვედა ფენებს უბრუნდება. ბალონს გამოიყენებენ მის ირგვლივ ტემპერატურის, წნევის, ტენიანობის, ქიმიური კონცენტრაციის და სხვა პარამეტრების გასაზომად. ბალონი რჩება გარკვეულ სიმაღლეზე რამდენიმე საათის, დღის ან წლების განმავლობაში, რათა შეკრიბოს მონაცემები გარემოს შესახებ. ბალონი დედამიწას უბრუნდება, თუ ჰელიუმი გაჟონავს ბალონიდან.

შესავალი

შესავალი

ამ თავში წარმოგიდგენთ MATLAB ფუნქციებს პოლინომის ანალიზისათვის. პირველ რიგში განვიხილავთ გზებს, როგორ შევქმნათ პოლინომი და როგორ ვაწარმოოთ გამოთვლები

პოლინომის საშუალებით. შემდეგ განვიხილავთ ამოცანას, სადაც ნაჩვენებია როგორ შევქმნათ მეტეოროლოგიური ბალონის სიჩქარის და სიმაღლის მოდელი პოლინომის საშუალებით. ამის შემდეგ განვიხილავთ ფუნქციებს, რომლებიც საშუალებას გვაძლევს ვიპოვოთ პოლინომის ფესვები და პირიქით ფესვების საშუალებით ავაგოთ პოლინომის გამოსახულება.

10.1 გამოთვლები პოლინომის საშუალებით.

პოლინომი არის ერთი ცვლადის ფუნქცია, რომლის ზოგადი ფორმულა ასეთია:

$$f(x) = a_1 x^N + a_2 x^{N-1} + a_3 x^{N-2} + \cdots + a_{N-1} x^2 + a_N x + a_{N+1}$$

სადაც ცვლადია x ხოლო კოეფიციენტები a_1, a_2, და ა.შ. პოლინომის ხარისხი ცვლადის ხარისხის ყველაზე მაღალი მაჩვენებლის ტოლია. ამრიგად, კუბური – მესამე ხარისხის პოლინომის ზოგადი ფორმულაა:

$$g(x) = a_0 x^3 + a_1 x^2 + a_2 x + a_3$$

მაგალითად:

$$h(x) = x^3 - 2x^2 + 0.5x - 6.5$$

პოლნომის კოეფიციენტის ინდექსი იწყება 1 –ით უმაღლესი ხარისხის მქონე ცვლადისათვის.

პოლინომს ხშირად ვიყენებთ საინჟინრო თუ სამეცნიერო ამოცანებში იმის გამო, რომ მათი საშუალებით შესაძლებელია კარგად აისახოს ფიზიკური სისტემები. მეცხრე თავში უკვე გავეცანით ექპერიმენტულ მონაცემთა მოდელირების მაგალითს პოლინომის საშუალებით.

9.1.1 პოლინომის მნიშვნელობის გამოთვლა (შეფასება)

MATLAB-ში არსებობს ცვლადის მოცემულ მნიშვნელობათათვის პოლინომის გამოთვლის რამდენიმე გზა. მაგალითად გვინდა გამოვთვალოთ შემდეგი პოლინომის მნიშვნელობა:

$$f(x) = 3x^4 - x^3 - 5.2$$

თუ გვინდა გამოვთვალოთ ამ ფუნქციის მნიშვნელობა სკალარისათვის, რომელიც ენიჭება x ცვლადს, დავწერთ:

f = 3*x^4 – 0.5*x^3 + x – 5.2

თუ x ვექტორია ან მატრიცა, მაშინ გამოვიყენებთ მასივურ ან შესაბამისი ელემენტების ოპერაციას:

f = 3*x.^4 – 0.5*x.^3 + x. – 5.2

მიღებული f მატრიცის ზომა ისეთივე იქნება, როგორც x მატრიცის.

პოლინომის მნიშვნელობა შეიძლება გამოვთვალოთ აგრეთვე ფუნქციით **polyval,** რომელსაც ორი არგუმენტი აქვს. პირველი არგუმენტი შეიცავს პოლინომის კოეფიციენტებს, მეორე კი x იმ მნიშვნელობებს, რომელთათვისაც ვითვლით პოლინომის მნიშვნელობებს. ამრიგად, f პოლინომის მნიშვნელობათა გამოსათვლელად დაგვჭირდება ბრძანებები:

```
a = [3, -0.5, 0, -5.2];
f = polyval(a,x);
```

ეს ორი ბრძანება შეიძლება გავაერთიანოთ:

```
f = polyval([3, -0.5, 0, -5.2], x);
```

f ზომა ისეთივე იქნება, როგორც x , რომელიც შეიძლება იყოს სკალარიც, ვექტორიც და მატრიცაც.

დავუშვათ გვინდა გამოვთვალოთ პოლინომის მნიშვნელობები ინტერვალში [0,5]:

$$g(x) = -x^5 + 3x^3 - 2.5x^2 - 2.5$$

შემდეგი ბრძანებები მოგვცემს პოლინომის 201 მნოშვნელობას ამ ინტერვალზე.

```
x = 0:5/200:5;
a = [-1, 0, 3, -2.5, 0, -2.5];
g = polyval(a,x);
```

როცა x ვექტორია ან სკალარი, ფუნქცია **polyval** პოლინომის მნიშვნელობას გამოითვლის შესაბამის ელემენტებს შორის ოპერაციით, როგორც წინა მაგალითშია აღწერილი. პოლინომოს მნიშვნელობები შეიძლება გამოვითვალოთ მატრიცული ოპერაციით. ამისათვის არსებობს ფუნქცია **polyvalm**:

f = polyvalm(a,X) – როცა a N+1 სიგრძის ვექტორია, რომელიც შეიცავს პოლინომის კოეფიციენტებს, გამოითვლის პოლინომის მნიშვნელობებს მატრიცული არგუმენტით X. X უნდა იყოს N x N ზომის კვადრატული მატრიცა. პოლინომის თავისუფალი წევრი გადამრავლდება ერთეულოვან მატრიცაზე I:

f = a(1)*X^N + a(2)*X^(N-1) + …+ a(N)*X*I

10.2 არითმეტიკული ოპერაციები

დავუშვათ ორი პოლინომის კოეფიციენტები ჩაწერილია სტრიქონი a და b ვექტორის სახით. შეგვიძლია ვაწაროოთ პოლინომური გამოთვლები. მაგალითად რომ შევკრიბოთ პოლინომები, უნდა შევკრიბოთ მათი შესაბამისი კოეფიციენტები. ამრიგად პოლინომების ჯამის კოეფიციენტები ტოლია შესაკრებ პოლინომთა შესაბამისი (ერთნაირი ხარისხის მქონე ცვლადის კიეფიციენტები) კოეფიციენტების ჯამისა. შევნიშნავთ, რომ ვექტორები, რომელიც შეიცავს პოლინომის კოეფეიციენტებს ერთნაირი სიგრძის უნდა იყოს. საილუსტრაციოდ დავუშვათ გვინდა შევკრიბოთ 2 პოლინომი;

$$g(x) = x^4 - 3x^2 - x + 2.4$$
$$h(x) = 4x^3 - 2x^2 + 5x - 16$$
$$s(x) = g(x) + h(x)$$

მათ შესაკრებად MATLAB –ში ვიყენებთ ბრძანებებს:

```
g = [1, 0, -3, -1, 2.4];
h = [0, 4, -2, 5, -16];
s = g + h;
```

მივიღებთ

```
s =

    1.0000    4.0000   -5.0000    4.0000  -13.6000
```

ასევე გამოითვლება პოლინომთა სხვაობაც. ორი პოლინომის სხვაობის ვექტორის ელემენტები მიიღება მათი ვექტირების შესაბამისი ელემენტების გამოკლებით. აქაც იგივე ბირობა უნდა იყოს დაცული – ვექტორები ერთნაირი სიგრძის უნდა იყოს.

პოლინომის სკალარულ სიდიდეზე გასამრავლებად მისი კოეფიციენტების ვექტორი უნდა გავამრავლოთ სკალარზე. ამრიგად, თუ გვაქვს პოლინომი $f(x) = 3x^2 - 6x + 1$, იმისათვის, რომ მივიღოთ $g(x) = 3f(x)$ პოლინომის კოეფიციენტები, ვსარგებლობთ ბრძანებებით:

```
f = [3, -6, 1];
g = 3*f;
```

სკლარის მნიშვნელობა შეიძლება დადებითიც იყო და უარყოფითიც.

პოლინომების ერთმანეთზე გადამრავლება უფრო რთული პროცესია, ვიდრე მათი შეკრება და გამოკლება. MATLAB პოლინომთა გადამრავლებას ახორციელებს ფუნქციით **conv**, რომელიც ასრულებს ნამრავლი პოლინომის კოეფიციენტების გამოთვლის ყველე საჭირო საფეხურს და გვაძლევს მისი შესაბამისი კიეფიციენტების ვექტირს. თუ a და b A და B პოლინომების ვექტორ-სტრიქონებია, C ნამრავლი პოლინომის კოეფიციენტები შემდეგი ბრძანებით გამოითვლება:

```
c = conv(a,b);
```

არ არის აუცილებელი, რომ a და b ვექტორები ერთნაირი სიგრძის იყოს.

პოლინომების გაყოფა MATLAB –ში ხორციელდება ბრძანებით **deconv**, რომელიც გვაძლევს ორ ვექტორს: პირველი ვექტორი შეიცავს განაყოფი პოლინომის კოეფიციენტებს, ხოლო მეორე ნაშთი პოლინომის კოეფიციენტებს. **deconv** და **conv** ფუნქციები ხშირად გამოიყენება სიგნალის დამუშავებში და დაწვრილებით განვიხილავთ ამ საკითხისადმი მიძღვნილ თავში.

conv და **deconv** ფუნქციათა საილუსტრაციოდ განვიხილოთ ორი პოლინომის ნამრავლი:

$$g(x) = (3x^3 - 5x^2 + 6x - 2)(x^5 + 3x^4 - x^2 + 2.5)$$

შეგვიძლია ეს ნამრვლი გამოვთვალოთ ბრძანებებით:

```
a = [3, -5, 6, -2];
b = [1, 3, 0, -1, 0, 2.5];
g =  conv(a,b);
```

g ვექტორი მიიღებს მნიშვნელობას – [3, 4, -9, 13, -1, 1.5, -10.5, 15, -5], რაც შეესაბამება პოლინომს:

$$g(x) = 3x^8 + 4x^7 - 9x^6 + 13x^5 - x^4 + 1.5x^3 - 10.5x^2 + 15x - 5$$

ფუნქცია **deconv** პოლინომთა გაყოფას ასრულებს. შევასრულოთ გაყოფა:

$$h(x) = \frac{3x^8 + 4x^7 - 9x^6 + 13x^5 - x^4 + 1.5x^3 - 10.5x^2 + 15x - 5}{x^5 + 3x^4 - x^2 + 2.5}$$

მივმართავთ ბრძანებებს:

```
n = [3, 4, -9, 13, -1, 1.5, -10.5, 15, -5];
d = [1, 3, 0, -1, 0, 2.5];
[q,r] =  deconv(n,d);
```

როგორც მოსალოდნელი იყო მივიღებთ განაყოფი პოლინომის კოეფიციენტებს [3, -5, 6, -2], ანუ პოლინომს $3x^3 - 5x^2 + 6x - 2$ და ნაშთის ვექტორს, რომელის ელემენტებიც ნულის ტოლია.

ახლა განვიხილოთ ასეთი განაყოფი:

$$h(x) = \frac{x^3 + 5x^2 + 11x + 13}{x^2 + 2x + 4}$$

MATLAB საშუალებით გამოვითვლით:

```
n = [1 5 11 13];
d = [1 2 4];
[q,r] = deconv(n,d);
```

მივიღებთ: q = [1 3] da r = [0 0 1 1].

რაც ნიშნავს:

$$h(x) = \frac{x^3 + 5x^2 + 11x + 13}{x^2 + 2x + 4} = x + 3 + \frac{x + 1}{x^2 + 2x + 4}$$

ხშირად საჭიროა, რომ პოლინომების განაყოფი წარმოვადგინოთ პოლინომური წილადების ჯამის სახით. MATLAB გააჩნია ფუნქცია **residue,** რომელიც საშუალებას გვაძლევს წილადი დავშალოთ რაციონალური წილადების ჯამად.

სავარჯიშო

მოცემულია პოლინომები:

$$f_1(x) = x^3 - 3x^2 - x + 3$$
$$f_2(x) = x^3 - 6x^2 + 12x - 8$$
$$f_3(x) = x^3 - 8x^2 + 20x - 16$$
$$f_4(x) = x^3 - 5x^2 + 7x - 3$$
$$f_5(x) = x - 2$$

ააგეთ შემდეგ ფუნქციათა მრუდი ინტერვალში [0,4]. გამოთვალეთ პოლინომის კოეფიციენტები ალგებრული ოპერაციებით(element by element) და MATLAB ფუნქციების (**conv, deconv, polyval**) გამოყენებით. შეადარეთ ერთმანეთს ორივე შემთხვევაში მიღებული შედეგების მიხედვით აგებული მრუდები.

1. $f_1(x)$
2. $f_2(x)$-2 $f_4(x)$
3. $3f_5(x)+ f_2(x)$-2 $f_3(x)$
4. $f_1(x)* f_3(x)$
5. $f_4(x)/(x$-1$)$
6. $f_1(x)* f_2(x)* f_5(x)$
7. $f_2(x)/(x$-1$)$

პრობლემა: მეტეოროლოგიური ბალონი

მეტეოროლოგიური ბალონის საშუალებით ხორციელდება ტემპერატურის და წნევის გაზომვა ატმოსფეროს სხვადასხვა სიმაღლეზე. ბალონი შევსებულია ჰელიუმით, რომლის სიმკვრივეც ბალონში ნაკლებია ბალონის ირგვლივ გარემოს სიმკვრივეზე ატმოსფეროს დაბალ ფენებში. ამიტომ იგი მიფრინავს სულ მაღლა და მაღლა, ვდრე არ მიაღწევს წონასწორობის წერტილს, ისეთ სიმაღლეს, სადაც ატმოსფეროს სიმკვრივე ბალონში ჰელიუმის სიმკვრივეს გაუტოლდება. დღის განმავლობაში მზე ათბობს რა ჰელიუმს, მისი სიმკვრივე ეცემა და ბალონი უფრო მაღალ ფენებში გადაადგილდება. ღამით ჰელიუმი ცივდება, იკუმშება და მისი სიმკვრივეც იზრდება, რის გამოც ბალონი ატმოსფეროს ქვედა ფენებში ეშვება. გარკვეული პერიოდის განმავლობაში გროვდება მონაცემები ბალონის სიმაღლის შესახებ. სიმაღლის ცვლილების მოდელირება შესაძლებელია პოლინომის საშუალებით.

დავუშვათ 48 საათის განმავლობაში ბალონის სიმაღლე დროის მიხედვით შემდეგი ფორმულის მიხედვით იცვლება:

$$h(t) = -0.12t^4 + 12t^3 - 380t^2 + 4100t + 220$$

სადაც დრო სათებშია მოცემული. ატმოსფეროში ბალონის გადაადგილების სიჩქარე ტოლი იქნება მისი წარმოებულის:

$$\upsilon(t) = -0.48t^3 + 36t^2 - 760t^2 + 4100$$

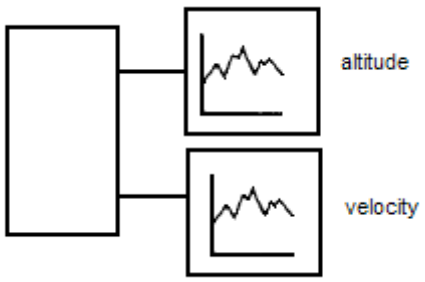
სიჩქარე გაზომილია მ/წმ. ააგეთ მეტეოროლოგიური ბალონის სიმაღლის და სიჩქარის მრუდები (მეტრი, წამი და მეტრი/წამში – ერთეულებში), განსაზღვრეთ ბალონის სიმაღლის უდიდესი მნიშვნელობა და სათანადო დროის მნიშვნელობა.

1. ამოცანის დასმა

მოცემული პოლინომების საშუალებით ააგეთ ატმოსფერული ბალონის სიმაღლისა და სიჩქარის გრაფიკი. იპოვეთ სიმაღლის უდიდესი და დროის სათანადო მნიშვნელობა.

2. INPUT/OUTPUT აღწერა

წარმოადგენს INPUT/OUTPUT დიაგრამას. როგორც ჩანს პროგრამას არ მიეწოდება არანაირი მონაცემები, ხოლო შედეგად იძლევა ორ მრუდს, სიმაღლის უდიდეს მნიშვნელობას და შესაბამის დროს.



ნახ. 10.1 I/O დიაგრამა

3. სახელდახელო ამოხსნა

ეს საფეხური აღარ გვჭირდება, რადგან უნდა დავწეროთ პროგრამა, რომელიც ააგებს მრუდს და იპოვის უდიდეს მნიშვნელობას მონაცემებს შორის, რომლის მიხედვითაც მრუდი აიგო.

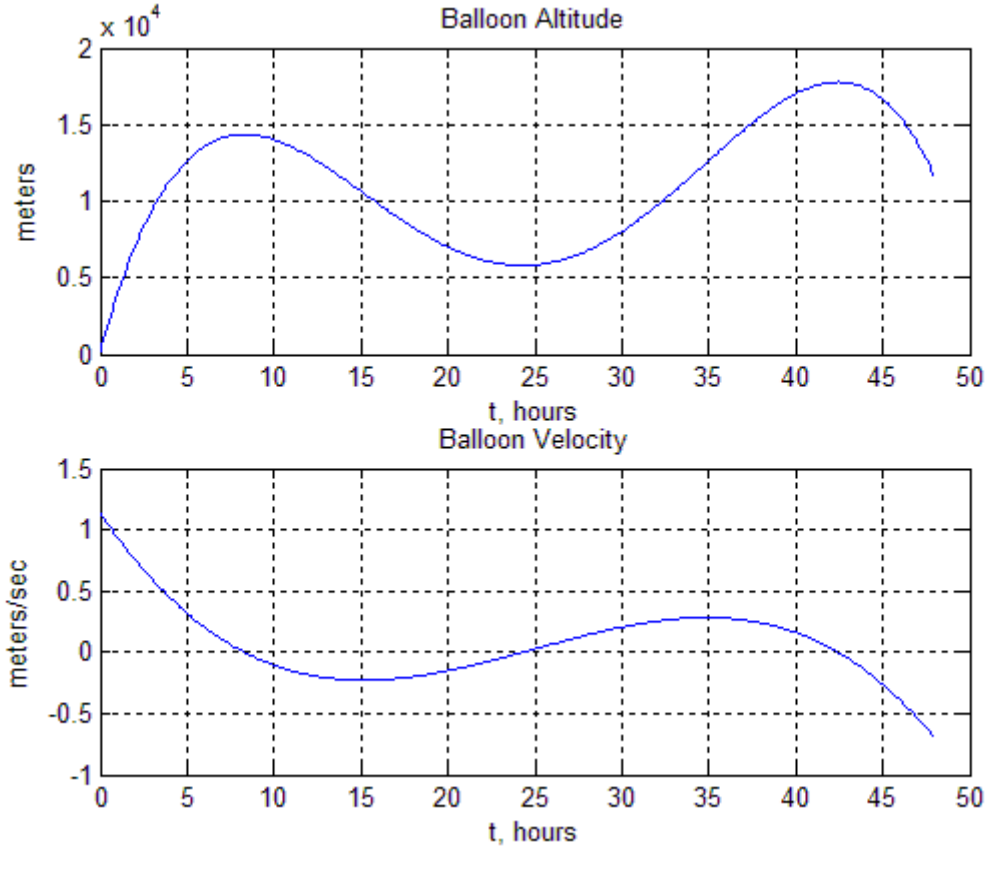
4. MATLAB ამოხსნა

```
%     This program generates altitude and velocity
%     plots using polinomial models for the
%     altitude and velocity of a weather ballon
%
t = 0:0.1:48;
alt_coef = [-0.12, 12, -380, 4100, 220];
vel_coef = [-0.48, 36, -760, 4100];
alt = polyval(alt_coef,t);
vel = polyval(vel_coef,t)/3600;
%
subplot(211),plot(t,alt),...
title('Balloon Altitude'),...
xlabel('t, hours'), ylabel('meters'),grid,...
subplot(212),plot(t,vel),...
title('Balloon Velocity'),...
xlabel('t, hours'), ylabel('meters/sec'),grid,pause
%
[max_alt,k] = max(alt);
max_time = t(k);
fprintf('Maximum altitude: %8.2f    Time: %6.2f \n',...
         max_alt, max_time)
```

5. შემოწმება

შედეგად მიღებული გრაფიკები ნაჩვენებია . მივიღებთ აგრეთვე სიმაღლის უდიდეს მნიშვნელობას:

```
Maximum altitude: 17778.57    Time:  42.40
```



ნახ. 10.2 მეტეოროლოგიური ბალონის სიმაღლე და სიჩქარე

10.3 პოლინომის ფესვები

ამოცანის ამოხსნისას ხშირად გვჭირდება ვიპოვოთ $y = f(x)$ განტოლების ფესვები, ანუ x ცვლადის ის მნიშვნელობები, რომელთათვისაც $y = 0$.

თუ f(x) ფუნქცია N რიგის პოლინომია, მას აქვს ზუსტად N ფესვი. მათ შორის შეიძლება იყოს ერთნაირი მნიშვნელობის მქონე ან კომპლექსური ფესვები. თუ დავუშვებთ, რომ პოლინომის კოეფიციენტები (a_1, a_2, . . .) ნამდვილი რიცხვებია და მას აქვს კომპლექსური ფესვები, ისინი ურთიერთშეუღლებული კომპლექსური რიცხვები იქნება.

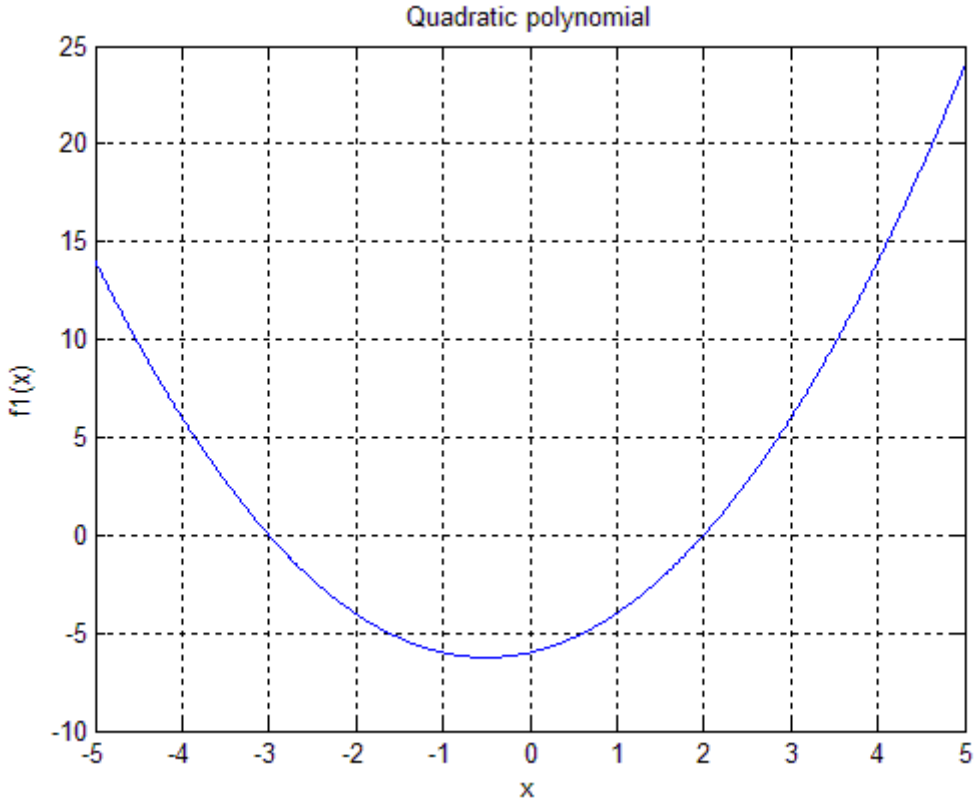
თუ პოლინომს დავშლით მამრავლებად ასეთი სახით:

$$f(x) = x^2 + x - 6 = (x-2)(x-3)$$

მისი ფესვების საპოვნელად $f(x)$ გავუტოლოთ 0, გვექნება:

$$(x-2)(x-3)=0$$

საიდანაც მივიღებთ ფესვების მნიშვნელობებს x = 2 და x = -3. ფესვები შეესაბამება x –ის იმ მნიშვნელობებს, რომელთათვისაც პოლინომის შესაბამისი მრუდი გადაკვეთს x ღერძს რაც ჩანს ნახ. 10.3-ზე.



ნახ. 10.3 კვადრატული პოლინომი 2 ნამდვილი ფესვით

კუბური პოლინომის ზოგადი სახეა:

$$f(x)=a_1x^3+a_2x^2+a_3x+a_4$$

კუბური პოლინომი მესამე ხარისხისაა, ამიტომ მას 3 ფესვი უნდა ჰქონდეს. თუ დავუშვებთ, რომ მისი კოეფიციენტები ნამდვილი რიცხვებია, შეიძლება გვქონდეს შემდეგი შემთხვევები:

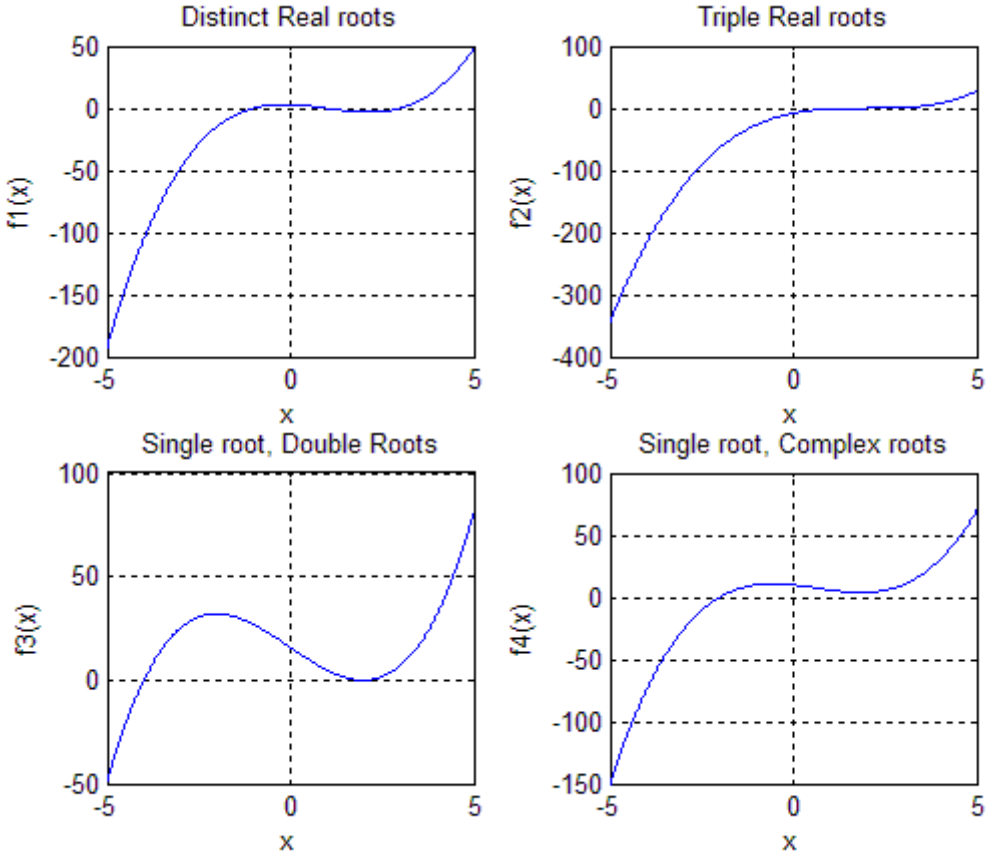
3 სხვადასხვა ნამდვილი ფესვი
3 ერთნაირი მნიშვნელობის მქონე ნამდვილი ფესვი,
2 ერთნაირი მნიშვნელობის მქონე და 1 განსხვავებული ნამდვილი ფესვი
1 ნამდვილი ფესვი და 2 ურთიერთშეუღლებული კომპლექსური ფესვი

საილუსტრაციოდ განვიხილოთ შემდეგი პოლიონომები:

$$f(x)=(x-3)(x+1)(x-1)=x^3-3x^2-x+3$$
$$f(x)=(x-2)^3=x^3-6x^2+12x-8$$
$$f(x)=(x+4)(x-2)^2=x^3-12x^2+16$$

$$f(x) = (x+2)(x-(2+j))(x-(2-j)) = x^3 - 2x^2 - 3x + 10$$

ნახ. 10.4 წარმოადგენს ამ პოლინომთა შესაბამის მრუდებს. ნემდვილი ფესვები x ის მნიშვნელობებია, სადაც ეს მრუდები x ღერძს გადაკვეთს.



ნახ. 10.4 კუბური პოლინომები

თუმცა მეორე და მესამე რიგის პოლინომების დაშლა მარტივ მამრავლებად და მათი ფესვების პოვნა შედარებით ადვილია, მაღალი რიგის პოლინომთა ფესვების პოვნა, საკმაოდ რთული და შრომატევადი საქმეა. არსებობს მრავალი რიცხვითი მეთოდი, რომელთა საშუალებით გამოითვლება პოლინომის ფესვები.

MATLAB პოლინომის ფესვების გამოსათვლელად გააჩნია ფუნქცია **roots.** მისი არგუმენტია ვექტორი, რომლის ელემენტებიც პოლინომის კოპეფიციენტებია. იგი გვაძლევს ვექტორს, რომლის ელემენტებიც პოლინომის ფესვებს წარმოადგენს. ფესვების რაოდენობა პოლინომის ხარისხის ტოლია. ამ ფუნქციის საილუსტრციოდ განვიხილოთ შემდეგი პოლინომი:

$$f(x) = x^3 - 2x^2 - 3x + 10$$

გამოვთვალოთ პოლინომის ფესვები MATLAB საშუალებით:

```
p = [1, -2, -3, 10];
r = roots(p)
```

ეს ორი ბრძანება ერთი ბრძანების სახითაც შეგვიძლია წარმოვადგინოთ:

```
r = root([1, -2, -3, 10])
```

MMATLAB მოგვცემს ფესვების მნიშვნელობებს: 2+i, 2-i, და –2. შევამოწმოთ მიღებული მნშვნელობებისშვის პოლინომი ღებულობს თუ არა ნულის ტოლ მნიშვნელობას:

```
polyval([1,-2,-3,10],r)
```

თუ მოცემული გვაქვს პოლინომის ფესვები და გვინდა ვიპოვოთ მისი კოეფიციენტების მნიშვნელობები, უნდა გამოვიყენოთ ფუნქცია **poly**. მისი არგუმენტია ვექტორი, რომელიც პოლინომის ფესვებს შეიცავს და შედეგად გვაძლევს ვექტორს, რომლის ელემენტებიც პოლინომის კოეფიციენტებია. მაგალითად, გამოვთვალოთ პოლინომის კოეფიციენტები, რომლის ფესვებია –1, 1, 3.

```
a = poly([-1, 1, 3]);
```

ვექტორ-სტრიქონი a = [1, -3, -1, 3].

სავარჯიშო

განსაზღვრეთ შემდეგი პოლონომის ფესვები. ააგეთ შესაბამისი მრუდი სათანადო ინტერვალში და შეამოწმეთ გადაკვეთს თუ არა მრუდი x ღერძს იმ წერილში, რომლის მნიშვნელობაც ფესვის ტოლია.

1. $g_1(x) = x^3 - 5x^2 + 2x + 8$
2. $g_2(x) = x^2 + 4x + 4$
3. $g_3(x) = x^2 - 2x + 2$
4. $g_4(x) = x^5 - 3x^4 - 11x^3 + 27x^2 + 10x - 24$
5. $g_5(x) = x^5 - 4x^4 - 9x^3 + 32x^2 + 28x - 48$
6. $g_6(x) = x^5 + 3x^4 - 4x^3 - 26x^2 - 40x - 24$
7. $g_7(x) = x^5 - 9x^4 + 35x^3 - 65x^2 + 64x - 26$
8. $g_8(x) = x^5 - 3x^4 + 4x^3 - 4x + 4$

ამ თავში განვიხილეთ პოლინომური ანალიზი. დავიწყეთ პოლინომის მნიშვნელობათა გამოთვლით, შემდეგ გავეცანით პოლინომების შეკრაბა-გამოკლებას და გამრავლება – გაყოფას. განვმარტეთ პოლინომის ფესვები და გავეცანით მათი პოვნის გზებს, განვიხილეთ ამ საკითხებთან დაკავშირებული MATLAB ფუნქციები.

10.4 ბრძანებები და ფუნქციები

conv	გადაამრავლებს ორ პოლინომს
deconv	გაყოფს ერთ პოლინომს მეორეზე
polyval	გამოითვლის პოლინომის მნიშვნელობას
roots	განსაზღვრავს პოლინომის ფესვებს
poly	გამოითვლის პოლინომის კოეფიციენტებს

პრობლემები

1-5 ამოცანა დაკავშირებულია ამ თავში განხილულ ამოცანებთან, 6-15 - ზოგადად პოლინომურ ანალიზს შეეხება

მეტეოროლოგიური ბალონი. ეს ამოცანები უკავშირდება ამ თავში განხილულ ამიცანას მეტეოროლოგიური ბალონის შესახებ.

1. ბალონის მოძრაობის მოდელზე დაყრდნობით განსაზღვრეთ დრო, როცა ბალონი დედამიწაზე დაეცემა (მითითება: განიხილე პოლინომის ფესვების მნიშვნელობები)
2. სიმაღლის მონაცემთა მიხდვით, რომელიც პოლინომის საშუალებით გამოითვლება, განსაზღვრეთ პერიოდი, როცა ბალონი გადაადგილდება ქვემოდან ზემოთ.
3. სიმაღლის მონაცემთა მიხდვით, რომელიც პოლინომის საშუალებით გამოითვლება, განსაზღვრე პერიოდი, როცა ბალონი ქვემოთ ეშვება.
4. ბალონის სიჩქარე ნულის ტოლია, როცა ის წყვეტს აღმასვლას, ან დაშვებას. გამოთვალეთ დროის მნიშვნელობა, როცა ბალონის სიჩქარე ნულის ტოლია
5. მე-4 ამოცანაში განვსაზღვრეთ წერტილი, სადაც სიჩქარე ნულის ტოილ იყო, რაც შეესაბამება იმ მომენტს, როცა ბალონი წყვეტს აღმასვლას ან დაშვებას. ეს წერტილები შეესაბამება იმ ინტერვალის საზღვრებს, რომლის განმავლობაშიც ბალონი ზემოთ მიფრინავს, ან ეშვება დედამიწისკენ. შეადარეთ მე-3 და მე-4 კითხვის პასუხები რომ ნახოთ კავშირი ბალონის სიჩქარესა და სიმაღლის მოკლება-მომატებას შორის.

პოლინომური ანალიზი. MATLAB საშუალებით იპოვეთ შემდეგი პრობლემების ამოხსნა

6. დავუშვათ მოცემულია ვექტორ-სტრიქონი, რომელიც შეიცავს პოლინომის კოეფიციენტებს. გამოთვალე და დაბეჭდე დადებითი ფესვების რაოდენობა
7. დავუშვათ მოცემულია სტრიქონი-ვექტორი, რომელიც შეიცავს პოლინომის კოეფიციენტებს. გამოთვალე ისეთი პოლინომის კოეფიციენტები, რომელსაც ექნება იგივე ნამდვილი ფესვები, მაგრამ არ ექნება კომპლექსური ფესვები.
8. დავუშვათ მოცემულია სტრიქონი-ვექტორი, რომელიც შეიცავს პოლინომის კოეფიციენტებს. გამოთვალე ისეთი ორი A(x) და B(x) პოლინომის კოეფიციენტები, რომელთა გადამრავლებითაც შესაძლებელია საწყისი პოლინომის მიღება, თანაც ისე, რომ A(x) ნამდვილი ფესვები ჰქონდეს, ხოლო B(x) კომპლექსური
9. იპოვეთ k-ს ისეთი მნიშვნელობები, რომელთათვისაც x-3 არის ერთ-ერთი თანამამრავლი პოლინომისა $kx^3 - 6x^2 + 2kx - 12$.
10. იპოვეთ k-ს ისეთი მნიშვნელობები, რომელთათვისაც x+2 არის ერთ-ერთი თანამამრავლი პოლინომისა $3x^3 + 2kx^2 - 4x - 8$.
11. შექმენით პოლინომი, რომლის კოეფიციენტები მთელი რიცხვებია ისე, რომ მისი ფესვები იყოს:
    (a) 1, 2, -3
    (b) 2/3. –2, -1
    (c) $\sqrt{5}, -\sqrt{5}, 4/3, 0$
    (d) ½, 2/3, 2i,-2i

12. შექმენით პოლინომი, რომლის ფესვი იქნება 3+ შემდეგი პოლინომის ფესვები $2x^3 - 7x + 5$
13. მოცემულია f(x) პოლინომის კოეფიციენტები. განსაზღვრე –f(x) პოლინომის კოეფიციენტები.
14. მოცემულია f(x) პოლინომის კოეფიციენტები. განსაზღვრეთ კოეფიციენტების ნიშნის ცვლილების რაოდენიობა (ჩვეულებრივ, მრავალწევრის წევრები დალაგებული უნდა იყოს ცვლადის ხარისხის კლების მიმართულებით მარცხნიდან მარჯვნივ) მაგალითად ასეთი კოეფიციენტების რაოდენობა პოლინომისათვის $2x^3 - 2x^2 + 2x - 3$ სამის ტოლია(+,-,+,-), ხოლო პლინომისათვის $x^2 + x - 2$ ორის (+,+,-).
15. დეკარტის ნიშნის ცვლილების წესი გვამცნობს, რომ თუ პოლინომის კოეფიციენტები ნამდვილი რიცხვებია, მისი დადებითი ფესვების რაოდენობა არ აღემატება ნიშნის ცვლილების რაოდენობას და უარყოფითი ფესვების რაოდენობა არ აღემატება ნიშნის ცვლილების რაოდენობას პოლინომისათვის *f(-x)*. მოცემულია პოლინომის კოეფიციენტები. იპოვეთ მისი დადებითი და უარყოფითი ფესვების რაოდენობა. (იხ. ამოცანა 13, 14.)

ალასკის ნავთობსადენი

11 რიცხვითი ინტეგრება და დიფერენცირება

პრობლემა: გაზის და ნავთობწარმოების გაუმჯობესება

ალასკის ნავთობსადენის აგებისას მრავალი პრობლემა წარმოიშვა. მათ შორის ერთ-ერთი უმნიშვნელოვანესი იყო ნიადაგის მრადიული ყინულით დაფარული საფარის დაცვა მილსადენის სითბოსაგან. მილში გამავალი ნავთობი თბება მილის კედლებზე ხახუნის გამო, ამიტომ საბჯენები, რომელსაც ეყრდნობა მილსადენი იზოლირებული უნდა იყოს ყინულოვანი ზედაპირისგან, ან მუდმივად ცივდებოდეს, რომ ყინული გალღობისაგან დაიცვას.

შესავალი

შესავალი

ინტეგრება და დიფერენცირება ძირითადი კონცეფციებია გამოთვლით მეთოდებში, რაც ფუნდამენტურ როლს თამაშობს მრავალი საინჟინრო და სამეცნიერო პრობლემის გადაჭრაში. თუ ზოგიერთი პრობლემა მოითხოვს ანალიზურ ამონახსნს ინტეგრებისა და დიფერენცირების საშუალებით, ზოგი მათგანის ამოხსნა შეუძლებელია ანალიზური მეთოდით და საჭირო ხდება რიცხვითი ინტეგრებისა და დიფერენცირების მეთოდის გამოყენება.

ამ თავში განვიხილავთ რიცხვითი ინტეგრაბისა და დიფერენცირების საკითხებს და MATLAB-ის შესაბამის ფუნქციებს.

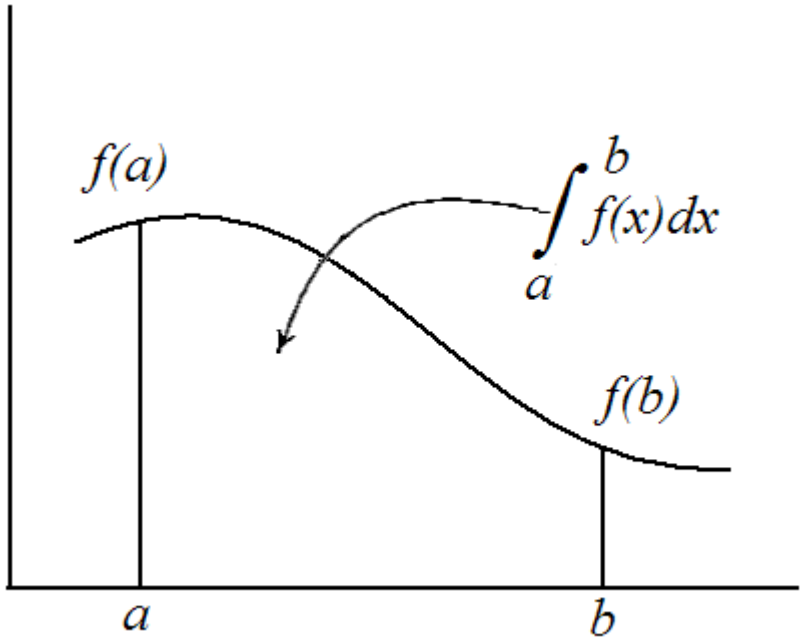
11.1 რიცხვითი ინტეგრება

f(x) ფუნქციის ინტეგრალი [a,b] ინტერვალზე განისაზღვრება როგორც ფართი, რომელსაც შემოსაზღვრავს *f(x)* წირი a-დან b-მდე. როგორც ნახ. 11.1 –ზეა ნაჩვენები თუ ინტეგრალის მნიშვნელობა K ტოლია, *f(x)* ფუნქციის საშუალებით იგი ასე ჩაიწერება:

$$K = \int_a^b f(x)$$

მრავალი ფუნქციისათვის ამ ინტეგრალის გამოთვლა შესაძლებელია ანალიზურად, მაგრამ ზოგიერთი ფიუნქციისათვის, როცა ეს შეუძლებელია, საჭიროა რიცხვით მეთოდებს მივმართოთ. ფუნქციის რიცხვითი ინტეგერების ამოცანა იმაში მდგომარეობს, რომ ინტეგრალქვეშა ფუნქციის მნიშვნელობათა მიხედვით გამოვთვალოთ ინტეგრალის მნიშვნელობა. ერთი ცვლადის ფუნქციის განსაზღვრული ინტეგრალის გამოთვლას ფუნქციის მნიშვნელობათა მიხედვით ზოგჯერ მექანიკურ კვადრატურას უწოდებენ. მექანიკური კვადრატურის ჩვეულებრივი ხერხი იმაში მდგომარეობს, რომ მოცემულ *f(x)* ფუნქციას განსახილველ ინტერვალზე ცვლიან მატივი სახის მაინტერპოლირებელი ან მააპროქსიმირებელი *g(x)* ფუნქციით (მაგალითად მრავალწევრით).

თუ ფუვქციას ძალიან მცირე ინტერვალებზე წრფის მონაკვეთებით შევცვლით, შეგვიძლია გამოვთვალოთ მიღებული ტრაპეციების ფართობები და შევკრიბოთ. ასე შევაფასებთ წირით შემოფარგლულ ფართს და ვიპოვით ინტეგრალის მნიშვნელობას. შეგვიძლია წრფის მაგიერ კვადრატული ფუნქცია ავიღოთ და ისე შევაფასოთ ინტეგრალის მნიშვნელობა. ამ მეთოდს სიმპსონის წესს უწოდებენ.



ნახ. 11.1 ფუნქციის ინტეგრალი

11.1.1 ტრაპეციის და სიმპსონის მეთოდი

თუ ინტერგრალქვეშა ფუნქციის შესაბამისი წირით შემოფარგლული ფართობი დაყოფილია მცირე ზომის ტრაპეციებად და თუ ინტერვალი [a,b] დაყოფილია n ტოლ ნაწილად მაშინ ფართობი აპროქსიმირდება შემდეგი ფორმულის საშუალებით (ტრაპეციის წესი):

$$K_T = \frac{b-a}{2n}(f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n))$$

სადაც x_i წარმოადგენს ტრაპეციის კიდურა წერტილებს და x_0 =a და x_n =b.
თუ წირით შემოფარგლული ფართი დაყოფილია კვადრატული წირით შემოფარგლულ ნაწილებად და ინტერვალი [a,b] დაყოფილია 2n ტოლ ნაწილად, მაშინ ფარათი აპროქსიმირდება სიმპსონის ფორმულით:

$$K_s = \frac{h}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 2f(x_{2n-2}) + 4f(x_{2n-1}) + f(x_n))$$

სადაც x_i წარმოადგენს დაყოფის წერტილებს და x_0 =a , x_{2n} =b და h=(b-a)/(2n).

თუ მააპროქსიმებელი ფუნქცია უფრო მაღალი რიგისაა (ტრაპეციის წესი ეყრდნობა წრფივ და სიმპსონის ფორმულა კვადრატულ ფუნქციას) ინტეგრალის მნიშვნელობის გამოსათვლელად სარგებლობენ სხვა მეთოდებით, მაგალითად - გაუს-ლობატოს მეთოდი.

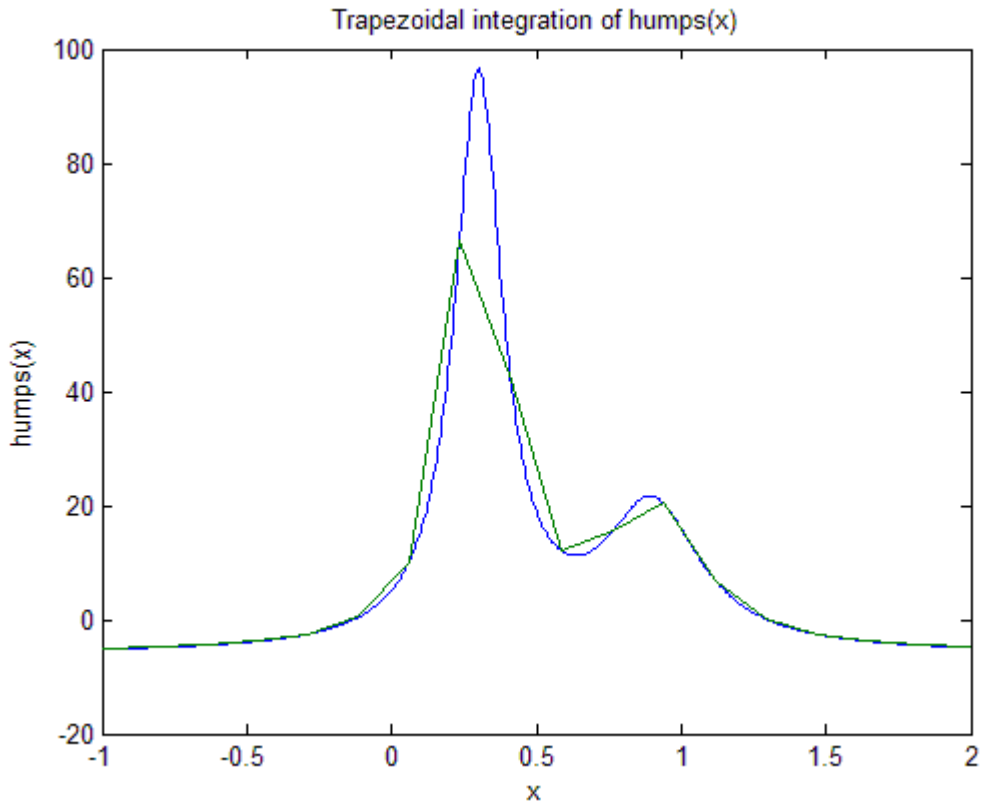
ინტეგრალის მნიშვნელობა უფრო ზუსტია, რაც უფო მცირე მონაკვეთებადაა დაყოფილი ინტერვალი. თუ შევეცდებით ფუნქციის ინტეგრებას (სინგულარობის) წყვეტის წერტილში (სადაც ფუნქცია ან მისი წარმოებული უსასრულობაა ან განსაზღვრული არ არის) დამაკმაყოფილებელ პასუხს ვერ მივიღებთ.

MATLAB ფუნქციები ინტეგრებისათვის

MATLAB -ს გააჩნია ფუნქცია ტრაპეციის წესით ინტეგრებისათვის:

trapz(x,y)	y ფუნქციის ინტეგრალი x-ის მიმართ ტრაპეციის წესით. x და y ერთნაირი სიგრძის ვექტორები უნდა იყოს,

ამ ფუნქციის საილუსტრაციოდ განვიხილოთ ფუნქცია humps(x), ეს MATLAB სადემონსტრაციო ფუნქციაა, აქვს პიკები x=0.3 და x=0.9 მახლობლად ნახ. 11.2.



ნახ. 11.2 ფუნქციის humps(x) გრაფიკი ინტერვალში [-1, 2]

trapz ახდენს ფართის აპროქსიმაციას ტრაპეციის წესით. n=18 ქვეინტერვალისათვის ინტეგრალის აპროქსიმირების პროცესი ასეთია:

```
x = linspace(-1, 2, 18)
y = humps(x);
area = trapz(x,y)

area =
   25.1406
```

თუ ნახაზის მიხედვით ვიმსჯელებთ ინტეგრალის არცთუ ისე კარგ შეფასებას მივიღებთ, მაგარმ თუ x უფრო მცირე ინტერვალებად დავყოფთ, შედეგსაც უკეთესს მივიღებთ.

```
x = linspace(-1,2,401);
y = humps(x);
area = trapz(x,y)
area =
26.3449
```

საინტერესოა ისეთი ინტეგრალის გამოთვლა, რომლის ზედა სზღვარი ცვლადია.

$$\int_a^x f(u)du$$

ინტეგრალის ქვედა საზღვარი განსაზღვრულია, x და y ერთნაირი ზომის ვექტორებია, cumtrapz(x,y) გვაძლევს y ფუნქციის კუმულაციურ ინტეგრალს, იმავე ზომის z ახალ ვექტორს, რომლის n-ური ელემენტი წარმოადგენს ინტეგრალის მნიშვნელობას x(1) დან x(n)-მდე (აპროქსიმირება ხორციელდება ტრაპეციის წესით).

MATLAB-ს ფუნქციის რიცხვითი ინტეგრებისათვის სხვა ფუნქციებიც აქვს. **quad** ფუნქცია ეყრდნობა სიმპსონის ადაპტირებულ ფორმულას, ხოლო **quadl** – გაუს - ლობატოს მეთოდს.

quad ფუნქციის უმარტივესი ფორმა მოითხოვს სამ არგუმენტს: ფუნქციის ბრჭყალებში ჩასმული სახელს, რომელიც გვაძლევს ვექტორს *f(x)* ფუნქციის მნიშვნელობებით, როცა მივაწვდით *x* მნიშვნელობათა ვექტორს; შეიძლება ეს იყოს MATLAB ფუნქციის სახელი, მაგალითად sin ან ჩვენს მიერ შექმნილი ფუნქციი სახელი. მეორე და მესამე არგუმენტი ინტერვალის a და b საზღვრებია, რომელშიც ინტეგრალის მნიშვნელობა გვინდა დავითვალოთ.

quad ფუნქციის საილუსტრაციოდ გამოვთვალოთ ინტეგრალი:

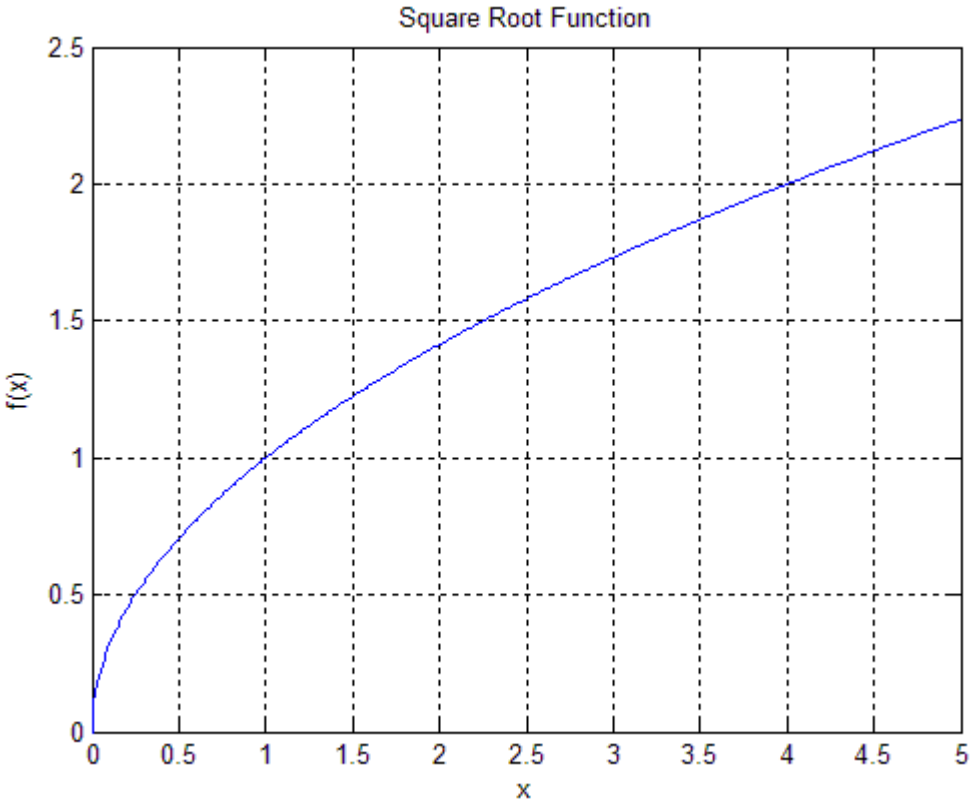
$$K_Q = \int_a^b \sqrt{x}\, dx$$

a და *b* არაუარყოფით მნიშვნელობათათვის.

$f(x) = \sqrt{x}$ ფუნქციის გრაფიკი ინტერვალში [0,5] წარმოდგენილია ნახაზზე; $x<0$ მნიშვნელობათათვის ფუნქციის მნიშვნელობები კომპლექსური რიცხვებია. ამ ფუნქციის ინტეგრება შესაძლებელია ანალიზურადაც და იგი ტოლია:

$$K = \frac{2}{3}\left(b^{3/2} - a^{3/2}\right)$$

შემდეგი პროგრამა საშუალებას გვაძლევს გამოვთვალოთ ინტეგრალის მნიშვნელობები განსაზღვრულ ინტერვალში **quad** ფუნქციით და ანალიზურად:



ნახ. 11.3 ფუნქცია $f(x) = \sqrt{x}$

```
%
%     This program compares the quad function with the
%     analytical results for the integration of the
%     square root of x over an interval [a, b], where
%     a and b are non-negative
%
a = input('Enter laft endpoint >= 0  ');
b = input('Enter right endpoint >= 0  ');
k = 2/3*(b^(1.5) - a^(1.5));
kq = quad('sqrt',a,b);
kq1 = quadl('sqrt',a,b);
fprintf('Analytical: %f\n Numerical: %f%f%f', k, kq, kq1)
```

შევამოწმოთ პროგრამა სხვადასხვა ინტერვალზე:

ინტერვალი [0.5, 0.6]

```
Analytical:   0.074136
Numerical:   0.074136  0.074136
```

ინტერვალი [0, 0.5]

```
Analytical:   0.235702
Numerical:   0.235695  0.235702
```

ინტერვალი [0, 1]

```
Analytical:   0.666667
Numerical:   0.666660  0.666665
```

quad და **quadl** ფუნქციებს შეიძლება ჰქონდეთ მეოთხე არგუმენტიც: დასაშვები მნიშვნელობა(tolerance) ფუნქციის ინტეგრება იქამდე გრძელდება, ვიდრე ცთომილება ამ პარამეტრზე ნაკლებია:

(წინა შეფასება - მიმდინარე შეფასება)/წინა შეფასება < დასაშვები მნიშვნელობა

თუ ეს პარამეტრი თავიდან არაა მითითებული, მის მნიშვნელობად ითვლება 0.001. თუ ამ ფუნქციებს აქვთ მეხუთე, არანულოვანი არგუმენტი, ინტეგრალის საბოლოო მნიშვნელობასთან ერთად შუალედური საფეხურის მნიშვნელობებსაც მოგვცემს.

თუ ფუნქციას გააჩნია სინგულარობის წერტილები შუალედის შიგნით, ინტერვალი რამდენიმე ქვეინტერვალად უნდა დაიყოს და ისე გამოითვალოს საბოლოო მნიშვნელობა, ხოლო სინგულარობის წერტილებში ინტეგრალის მნიშვნელობა შეფასდეს სხვა მეთოდით, მაგალითად ლოპიტალის წესით.

სავარჯიშო

მოცემულია ფუნქცია $f(x) = |x|$, გამოთვალეთ შემდეგი ინტეგრალი ანალიზურად და **quad** ფუნქციის საშუალებით. შედეგები შეადარეთ ერთმანეთს:

1. $\int_{0.5}^{0.6} |x|\,dx$
2. $\int_{0.0}^{1} |x|\,dx$
3. $\int_{-1}^{-0.5} |x|\,dx$
4. $\int_{-1}^{0} |x|\,dx$
5. $\int_{-0.5}^{0.5} |x|\,dx$

პრობლემა: მილსადენში ნავთობის ნაკადის მოძრაობის ანალიზი

ამ მაგალითში გავაანალიზებთ მილსადენში ნავთობის ნაკადის დინებას. წრიულ მილში სითხის დინების ანალიზს გამოყენების მრავალი სფერო გააჩნია, მათ შორის სისხლის დინება არტერიებსა და ვენებში, წყალმომარაგების სისტემა ქალაქში, ირიგაციული სისტემა სოფლის მეურნეობაში, მელნის ნაკადი ჭავლური ტიპის პრინტერში და სხვა.

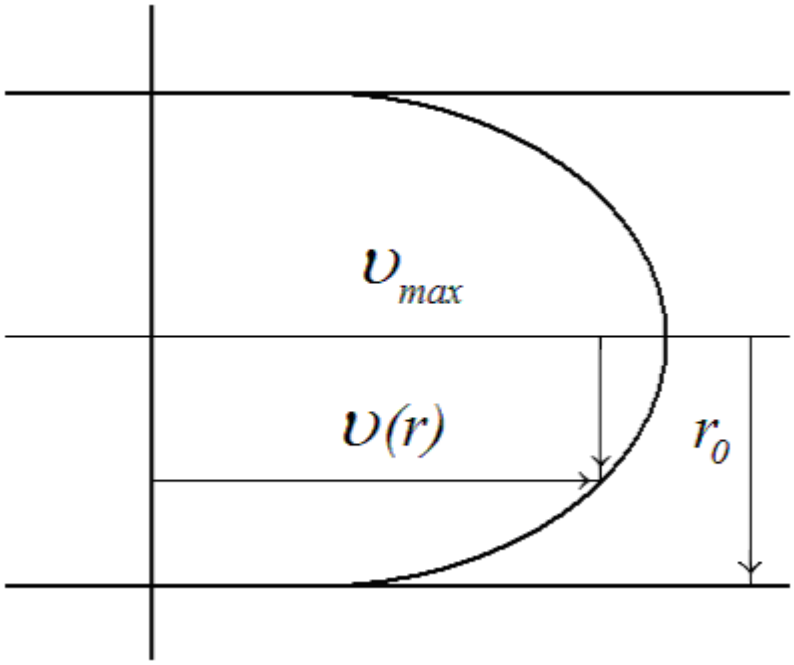
წრიულ მილში სითხის დინებისას ხახუნი მილის კედლებზე განაპირობებს 'სიჩქარის პროფილს'. სითხის ის ნაწილი, რომელიც უშუალოდ ეხება მილის კედლებს, თითქმის არ მოძრაობს, ხოლო მილის ცენტრში გამავალი სითხე ყველაზე სწრაფად მოძრაობს. ნახ. 11.4 მოცემული დიაგრამა გვიჩვენებს როგორ იცვლება მილში გამდინარე ნავთობის სიჩქარე მილის დიამეტრის გასწვრივ და განსაზღვრავს ანალიზისათვის საჭირო ცვლადებს. სიჩქარის პროფილს აღწერს შემდეგი განტოლება:

$$\upsilon(r) = \upsilon_{\max}\left(1 - \frac{r}{r_0}\right)^{1/n}$$

n მთელი რიცხვია 5 -სა და 10 -ს შორის, რომელიც განსაზღვრავს ნაკადის ფორმას (forward flow). ჩვენს შემთხვევაში n=8. მილში ნაკადის საშუალო სიჩქარე გამოითვლება სიჩქარის პროფილის ინტეგრებით 0-დან მილის რადიუსის r_0 მნიშვნელობამდე, ფორმულით:

$$\upsilon_{ave} = \frac{\int_0^{r_0} \upsilon(r) 2\pi r\, dr}{\pi r_0^2} = \frac{2\upsilon_{\max}}{r_0^2}\int_0^{r_0} r\left(1 - \frac{r}{r_0}\right)^{1/n} dr$$

n და $\upsilon_{\max}$ მნიშვნელობები შესაძლებელია გაიზომოს ექსპერიმენტულად, r_0 მილის რადიუსია. დაწერეთ MATLAB პროგრამა, რომელიც სიჩქარის პროფილის ინტეგრებით განსაზღვრავს მილში ნავთობის ნაკადის მოძრაობის საშუალო სიჩქარეს.



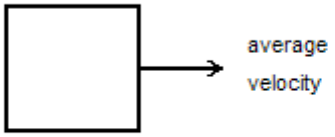
ნახ. 11.4 მილში გამდინარე ნავთობის სიჩქარის პროფილი

1. ამოცანის დასმა

გამოვთვალოთ მილში გამდინარე ნავთობის საშუალო სიჩქარე.

2. INPUT/OUTPUT აღწერა

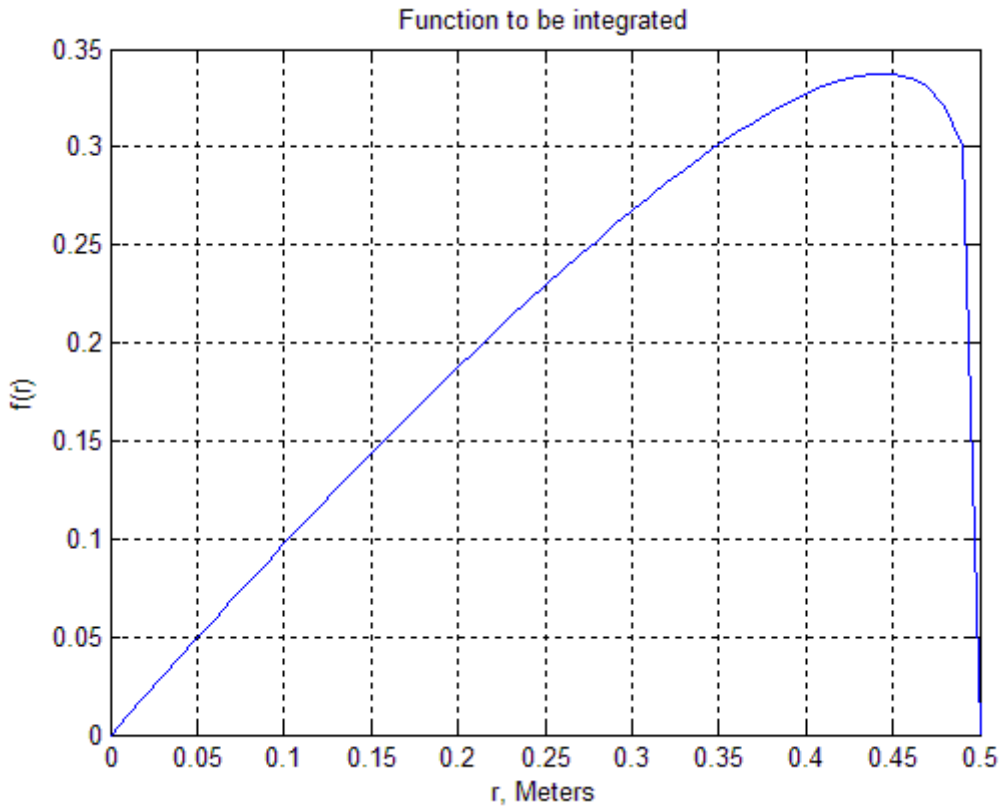
როგორც ნახ. 11.5 დიაგრამა უჩვენებს, პროგრამამ შედეგად უნდა მოგვცეს მილში გამდინარე ნავთობის ნაკადის საშუალო სიჩქარე. ნაკადის სიჩქარის უდიდესი მნიშვნელობა $\upsilon_{\max}$, მილის რადიუსი r_0 და n პროგრამაში განსაზღვრულია როგორც მუდმივი სიდიდეები.



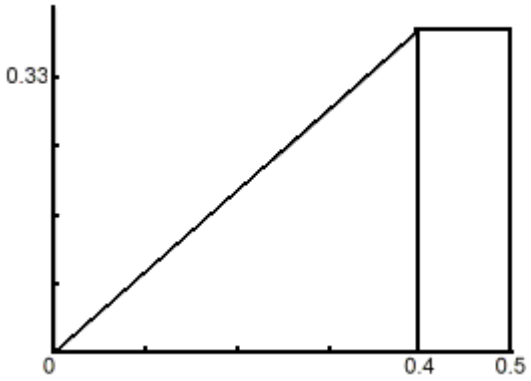
ნახ. 11.5 I/O დიაგრამა

3. სახელდახელო ამოხსნა

თუ დავუშვებთ, რომ r_0=0,5 მ, n=8, შეგვიძლია ავაგოთ $r(1-r/r_0)^{1/n}$ ფუნქციის გრაფიკი, რომელიც ნაჩვენებია ნახ. 11.6 ამ ფუნქციის ინტეგრალის საშუალო მნიშვნელობა შეგვიძლია შევაფასოთ ამ ნახაზიდან იმ სამკუთხედებისა და მართკუთხედების ჯამის საშუალებით(ნახ. 11.7), რომელთაც შემოწერს ფუნქციის გრაფიკი:



ნახ. 11.6 მილში გამდინარე ნავთობის ნაკადის სიჩქარის პროფილი



ნახ. 11.7 ინტეგრალის აპროქსიმირება

ფართობი = $0.5(0.4)(0.35)=0.105$

გავამრავლოთ მიღებული ფართობი $2\upsilon_{max}/r_0^2$, რომ მივიღოთ სიჩქარის საშუალო მნიშვნელობა. თუ დავუშვებთ, რომ υ_{max} = 1.5, მივიღებთ სიჩქარის საშუალო მნიშვნელობას 1.260.

4. MATLAB ამოხსნა

ინტეგრალის გამოსათვლელად ვისარგებლოთ **quad** ფუნქციით. მისი ერთ-ერთი პარამეტრია იმ ფუნქციის სახელი, რომელიც გამოითვლის ინტეგრალქვეშა ფუნქციის მნიშვნელობებს. ამიტომ უნდა დავწეროთ M-ფუნქცია, რომელიც გამოითვლის ინტეგრალქვეშა ფუნქციის მნიშვნელობებს. პროგრამაშიც და ფუნქციაშიც მუდმივების სახით უნდა შევიტანოთ შემდეგი სიდიდეების მნიშვნელობები: υ_{max}, r_0.

```
%
%     This program computes the value of the
%     averaje flow velosuty for a pipeline
%     using numerical integration.
%
vmax = 1.5;
r0 = 0.5;
%
integral = quad('velocity', 0, 0.5)
%
ave_velocity = (2*vmax/(r0^2))*integral
```

უნდა შევქმნათ აგრეთვე M-ფაილი velocity.m რომელიც განსაზღვრავს ინტეგრალქვეშა ფუნქციას:

```
function v = velocity(r)
%     VELOCITY     This function is related to the
%     average flow velocity of the pipe.
%
r0 = 0.5;
```

```
n = 8;
%
v = r.*(1-r/r0).^(1/n);
```

5. შემოწმება

შედეგად მივიღებთ :

```
integral =
          0.1046
ave_velocity =
          1.2548
```

ჩვენს მიერ შეფასებული მნიშვნელობები იყო: 0.105 და 1.260, შესაბამისად.

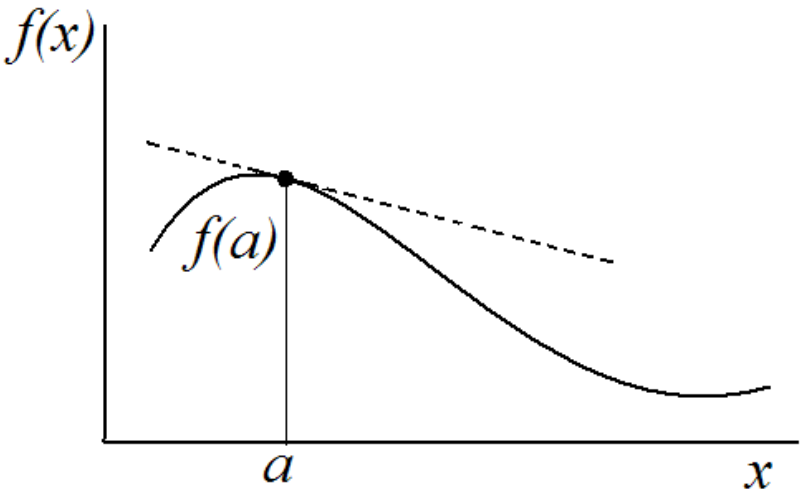
11.2 რიცხვითი დიფერენცირება

ფუნქციის *f(x)* წარმოებული არის ახალი ფუნქცია *f′(x)* რომელიც განისაზღვრება როგორც *x* არგუმენტის ცვლილების სიჩქარე. იგი განისაზღვრება როგორც ფუნქციის ნაზრდის თანაფარდობა არგუმენტის ნაზრდთან.

$$f'(x) = \frac{df(x)}{dx}$$

მრავალი ფიზიკური პროცესის შესწავლისას გვჭირდება შევაფასოთ ცვლადის ცვლილების სიჩქარე. მაგალითად, სიჩქარე მდებარეობის ცვლილების ზომაა (მ/წმ), ხოლო აჩქარება - თავად სიჩქარის ცვლილების (მ/წმ^2). შეიძლება ვაჩვენოთ, რომ აჩქარების ინტეგრალი სიჩქარის ტოლია, ხოლო სიჩქარის ინტეგრალი - მანძილის. ამრიგად ინტეგრება და დიფერენცირება ურთიერთშებრუნებული პროცესებია, ფუნქციის წარმოებულის ინტეგრება საწყის ფუნქციას გვაძლევს. ინტეგრალის წარმოებული იძლევა საწყის ფუნქციას მუდმივი სიდიდის სხვაობით.

გეომეტრიულად ფუნქციის წარმოებული წარმოადგენს მოცემულ წერტილში ფუნქციის მხების მიერ x ღერძთან შედგენილი კუთხის ტანგენსს. a წერტილში ფუნქციის წარმოებულური *f(a)* ნაჩვენებია ნახ. 11.8.



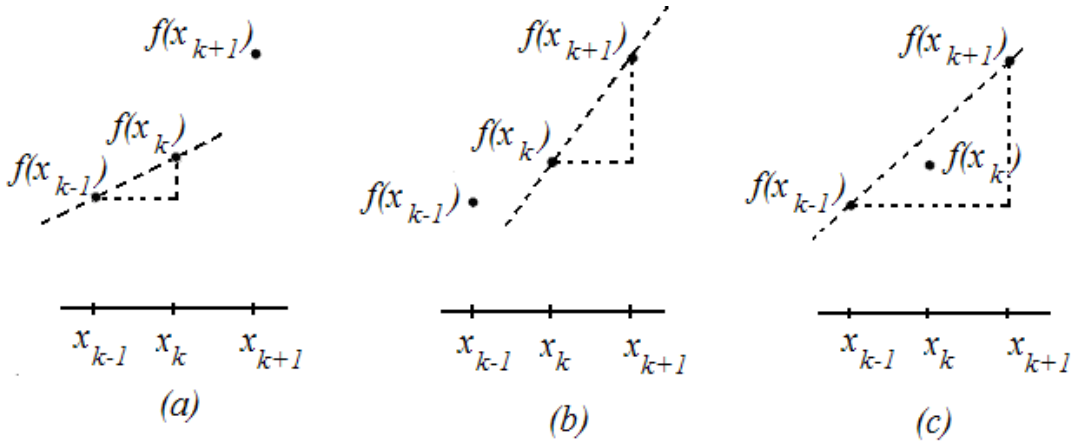
ნახ. 11.8 *f(x)* ფუნქციის წარმოებული a წერტილში

რადგან ფუნქციის წარმოებული მოცემულ წერტილში ამ წერტილზე გავლებული მხების დახრაა, თუ ვიცით, რომ ფუნქციის წარმოებული მოცემულ წერტილში ნულის ტოლია, მაშინ ამ წერტილში გავლებული მხები ჰორიზონტალური წრფე იქნება. ფუნქციის იმ წერტილებს, სადაც წარმოებული ნულის ტოლია ფუნქციის კრიტიკულ წერტილებს უწოდებენ და ეს შეიძლება იყოს ფუნქციის ჰორიზონტალური არე, ლოკალური მაქსიმუმი ან მინიმუმი. (შეიძლება ასეთი წერტილი გლობალური მაქსიმუმი ან მინიმუმიც იყოს, ამის დასადგენად ფუნქციის შემდგომი ანალიზია საჭირო). თუ ფუნქციას გავაწარმოებთ რაიმე ინტერვალში და ვნახავთ, რომ წარმოებულის ნიშანი იცვლება, ეს იმიზე მიუთითებს, რომ ფუნქციას ამ ინტერვალში ლოკალური მინიმუმი ან მაქსიმუმი გააჩნია. იმის შესაფასებლად მაქსიმუმთან გვაქვს საქმე თუ მინიმუმთან მეორე წარმოებულს უნდა მივმართოთ(მეორე წარმოებული მიიღება ფუნქციის წარმოებულის გაწარმოების შედეგად). თუ მოცემულ წერტილში ფუნქციის მეორე წარმოებული დადებითია, მაშინ ფუნქციის მნიშვნელობა მოცემულ წერტილში – ლოკალური მინიმუმია და პირიქით, თუ მეორე წარმოებული მოცემულ წერტილში უარყოფითია, ამ წერტილში ფუნქციას მაქსიმუმი გააჩნია.

11.2.1 დიფერენცირება

რიცხვითი დიფერენცირების მეთოდი განსაზღვრავს მოცემულ x_k წერტილში ფუნქციის წარმოებულის მნიშვნელობას ამ წერტილში გავლებული მხების აპროქსიმირებით x_k წერტილში ფუნქციის მნიშვნელობათა გამოყენებით. აპროქსიმირება შეიძლება მოხდეს რამდენიმე გზით. შევხედოთ ნახ. 11.9, ნახ. 11.9(*a*) მიუთითებს, რომ წარმოებული x_k წერტილში შეფასდება $f(x_{k-1})$ და $f(x_k)$ წერტილებზე გავლებული წრფის დახრით:

$$f'(x) = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$



ნახ. 11.9 *f(x)* ფუნქციის წარმოებულის გამოთვლის სხვადასხვა ხერხი

ამგვარად შეფასებულ წარმოებულს მარცხენა უწოდებენ. ნახ. 11.9*(b)* უჩვენებს, რომ აპროქსიმირება ხდება წრფის დახრით, რომელიც გაივლის წერტილებზე $f(x_{k+1})$ და $f(x_k)$:

$$f'(x) = \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}$$

ამგვარად შეფასებულ წარმოებულს მარჯვენა დიფერენციალურ აპროქსიმირებას უწოდებენ. ჩვეულებრივ იგულისხმება, რომ x_k მდებარეობს x_{k-1}-სა და x_{k+1} შორის. ამ ორივე შემთხვევაში წარმოებულის შეფასების სიზუსტე დამოკიდებულია წერტილებს შორის მანძილზე. წარმოებულის გამოთვლის მაღალი სიზუსტე მიიღწევა წერტილებს შორის მანძილების შემცირებით.

f(x) ფუნქციის მეორე რიგის წარმოებული მისი წარმოებულის წარმოებულს ეწოდება:

$$f''(x) = \frac{df'(x)}{dx}$$

ეს ფუნქცია შეფასდება პირველი წარმოებულის საშუალებით. ამრიგად თუ გამოვიყენებთ მარცხენა დიფერენცირებას, მივიღებთ:

$$f''(x) = \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k=1}}$$

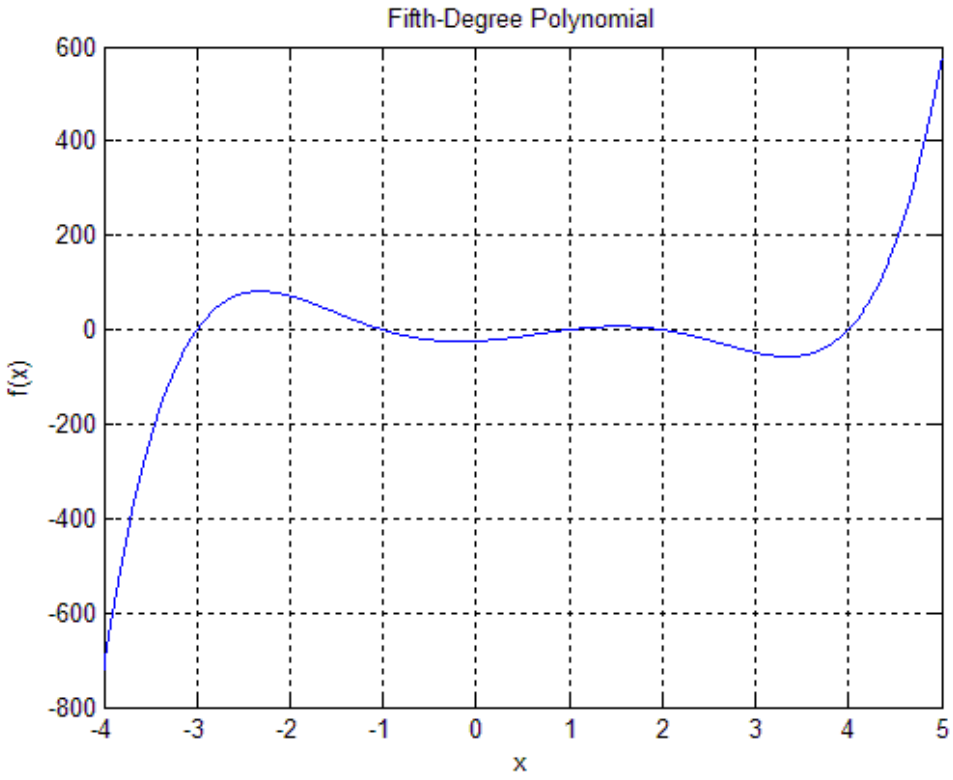
diff ფუნქცია გამოითვლის სხვაობებს ვექტორში ურთიერთმომდევნო ელემენტებს შორის და გვაძლევს ახალ ვექტორს, რომლის ელემენტების რაოდენობა 1-ით ნაკლებია. თუ მისი არგუმენტი მატრიცაა, აღწერილ ოპერაციას ასრულებს მატრიცის თითოეული სვეტისათვის). დავუშვათ ვექტორი x შეიცავს ელემენტებს: [0, 1, 2, 3, 4, 5], ხოლო y –[2, 3, 1, 5, 8, 10], მაშინ diff(x) = [1, 1, 1, 1, 1], diff(y) = [1, -2, 4, 3, 2] y ფუნქციის წარმოებული x-ით იქნება dy = diff(y)./diff(x). ყურადღება მიაქციეთ, რომ წარმოებულის ეს მნიშვნელობები სწორ შედეგს იძლევა როგორც მარცხენა, ისე მარჯვენა დიფერენცირების ფორმულებისათვის. განსხვავება ორი სხვადასხვა მეთოდით გამოთვლილ წარმოებულს შორის განისაზღვრება x ვექტორის მნიშვნელობებით, რომელიც შეესაბამება წარმოებულს dy. თუ x შესაბამისი მნიშვნელობებია [1, 2, 3, 4, 5], მაშინ dy გამოითვლის მარცხენა დიფერენციალს,

თუ x შესაბამისი მნიშვნელობებია [0, 1, 2, 3, 4], მაშინ dy გამოითვლის მარჯვენა დიფერენციალს.

დავუშვათ მოცემულია ფუნქცია:

$$f(x) = x^5 - 3x^4 - 11x^3 + 27x^2 + 10x - 24$$

ამ ფუნქციის გრაფიკი ნაჩვენებია ნახ. 11.10. დავუშვათ გვინდა გამოვითვალოთ ამ ფუნქციის წარმოებული ინტერვალში [-4, 5], მარცხენა დიფერენცირების ხერხით. შეგვიძლია გამოვიყენოთ ფუნქცია diff. df წარმოადგენს f′(x), ხოლო xd x-ის შესაბამის მნიშვნელობებს.



ნახ. 11.10 მეხუთე ხარისხის პოლინომის მრუდი

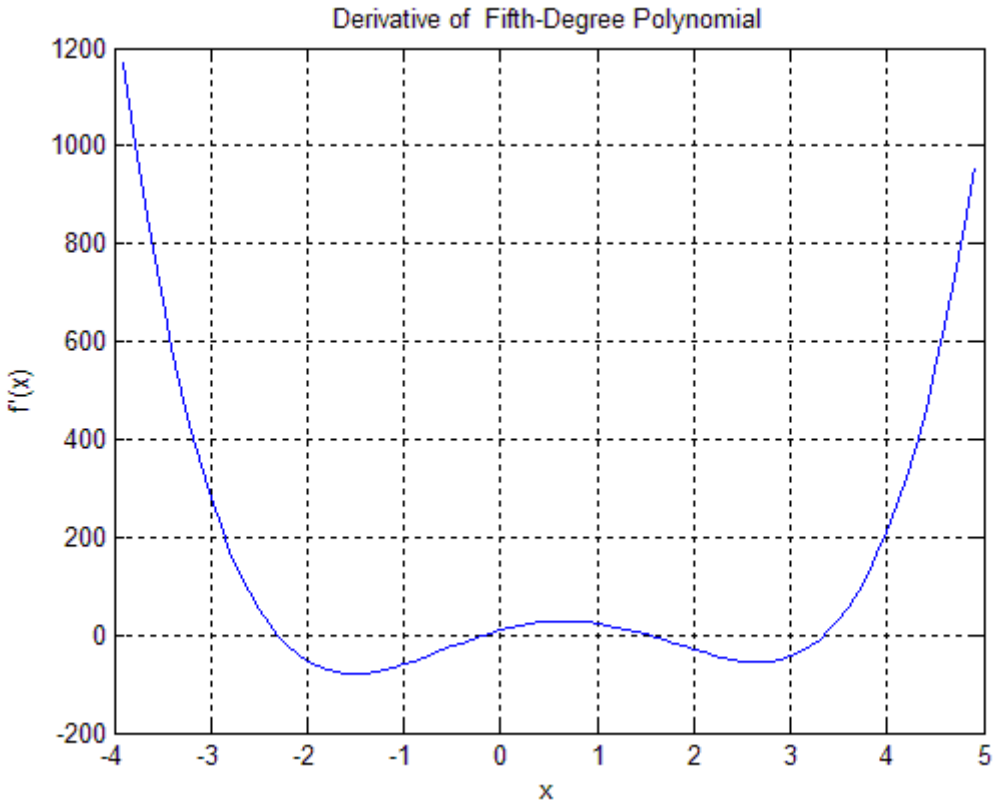
```
x=-4:0.1:5;
f = x.^5 -3*x.^4 - 11*x.^3 + 27*x.^2 + 10*x -24;
df = diff(f)./diff(x);
xd = x(2:length(x));
```

ნახ. 11.11 წარმოადგენს წარმოებულის გრაფიკს. დააკვირდით, წერტილები, სადაც ფუნქციის წარმოებული ნულს უტოლდება ფუნქციის ლოკალურ მინიმუმსა და მაქსიმუმს შეესაბამება. ამ ფუნქციას არ გააჩნია გლობალური მინიმუმი და მაქსიმუმი, იმიტომ რომ იგი ვრცელდება -∞ -დან +∞ -მდე. შეგვიძლია ვიპოვოთ ამ ფუნქციის კრიტიკულ მნიშვნელობათა მდებარეობა (x = -2.3, -0.2, 1.5, 3.4) შემდეგი ბრძანებებით:

```
product = df(1:length(df)-1).*df(2:length(df));
critical =xd(find(product<0))
```

ამ გამოსახულებაში ფუნქცია **find** განსაზღვრავს იმ ელემენტის ინდექსს ვექტორში **product**, რომლის შესაბამისი df(k) ნულის ტოლია. შემდეგ ეს ინდექსი გამოყენებულია xd ვექტორის

იმ ელემენტების საპოვნელად, რომელიც კრიტიკულ წერტილს შეესაბამება (x-ის იმ მნიშვნელობებს, სადაც ფუნქციის წარმოებული ნულის ტოლია).



ნახ. 11.11 მეხუთე ხარისხის პოლინომის წარმოებულის მრუდი

ცენტრალურ წარმოებულს x და f ვექტორებზე დაყრდნობით შემდეგი ბრძანებების საშუალებით ვიპოვით:

```
numerator = f(3 : length(f)) – f(1 : length(f) – 2);
denomirator = x(3 : length(x)) – x(1 : length(x) – 2);
dy = numerator./denomirator;
xd = x(2 : length(x) – 1);
```

ჩვენს მიერ განხილულ მაგალითში დავუშვით, რომ ფუნქცია წარმოებადია, მაგრამ მრავალ ამოცანაში მოცემული გვაქვს ექსპერიმენტული მონაცემები და შეუძლებელია ავირჩიოთ წერტილები ერთმანეთთან ისე ახლოს, რომ წარმოებულის მნიშვნელობები ზუსტად განვსაზღვროთ. ასეთ შემთხვევაში მივმართავთ მე-9 თავში აღწერილ მეთოდს. ვიპოვით მრავალწევრს, რომელიც ამ მონაცემებს შეესაბამება და ისე გამოვითვლით მიღებული ფუნქციის წარმოებულს.

სავარჯიშო

თითოეული გამოსახულებისათვის ააგეთ ფუნქციის, მისი პირველი წარმოებულის და მისი მეორე წარმოებულის გრაფიკი ინტერვალში [-10,10]. შემდეგ MATLAB

ბრძანებების საშუალებით იპოვეთ და დაბეჭდეთ ლოკალური მინიმუმის, ლოკალური მაქსიმუმის და x -ის შესაბამისი მნიშვნელობები.

1. $g_1(x) = x^3 - 5x^2 + 2x + 8$
2. $g_2(x) = x^2 + 4x + 4$
3. $g_3(x) = x^2 - 2x + 2$
4. $g_4(x) = 10x - 24$
5. $g_5(x) = x^5 - 4x^4 - 9x^3 + 32x^2 + 28x - 48$
6. $g_6(x) = x^5 + 3x^4 - 4x^3 - 26x^2 - 40x - 24$
7. $g_7(x) = x^5 - 9x^4 + 35x^3 - 65x^2 + 64x - 26$
8. $g_8(x) = x^5 - 3x^4 + 4x^3 + 32x^2 - 4x + 4$

დასკვნა

ამ თავში განვიხილეთ რიცხვითი ინტეგრება და დიფერენცირება. რიცხვითი ინტეგრაბა ხდება წირით შემოფარგლული ფართის აპროქსიმირებით, ხოლო დიფერენცირება ფუნქციის მრუდის მოცემულ წერტილზე გავლებული მხების აპროქსიმირებით. გავეცანით MATLAB ფუნქციებს ინტეგრებისათვის - **quad** და **quadl,** რომლებიც ეყრდნობა სიმპსონის და გაუს - ლობატოს მეთოდებს შესაბამისად. ორივე ფუნქციისათვის საჭიროა, რომ ფუნქცია იყოს ან MATLAB – ის ძირითადი ფუნქცია, ან ჩვენს მიერ M-ფაილის სახით წარმოდგენილი MATLAB ფუნქცია. წარმოებულის გამოთვლას MATLAB –ში ემსახურება ფუნქცია **diff,** რომელიც გამოითვლის სხვაობებს ვექტორის მეზობელ ელემენტებს შორის. გამოითვლება ამგვარი სხვაობები როგორც არგუმენტის, ისე ფუნქციისათვის და მივიღებთ წარმოებულს: diff(f)./diff(x).

ბრძანებები და ფუნქციები

trapz	გამოითვლის ინტეგრალს ტრაპეციის წესით
cumtrapz	გვაძლევს ფუნქციის კუმულაციურ ინტეგრალს
diff	გამოითვლის სხვაობებს ვექტორის მეზობელ ელემენტებს შორის
quad	გამოითვლის ინტეგრალს (სიმპსონის წესით)
quadl	გამოითვლის ინტეგრალს (გაუს-ლობატოს წესით)

ამოცანები

1-4 ამოცანა დაკავშირებულია ამ თავში განხილულ ამოცანასთან, ხოლო 5 – 11 ახალ პრობლემებს უკავშირდება.

მილსადენში სითხის ნაკადის ანალიზი.

1. ამ პრობლემაში აღწერილი პარამეტრების საშუალებით ააგეთ სიჩქარის პროფილი.
2. შეადგინეთ ცხრილი, რომელიც უჩვენებს სითხის ნაკადის საშუალო სიჩქარეს n სიდიდის სხვადასხვა მნიშვნელობებისათვის 5-დან 10-მდე.

3. შეადგინეთ ცხრილი, რომელიც უჩვენებს ნავთობის ნაკადის საშუალო სიჩქარეს მილის რადიუსის სხვადასხვა მნიშვნელობათათვის: 0.5, 1.0, 1.5 და 2.0. სხვა პარამეტრები უცვლელად დატოვეთ.
4. შეცვალეთ პროგრამა ისე, რომ საშუალება გქონდეს მივაწოდოთ პროგრამას v_{max} მნიშვნელობა.

ტრაექტორიის მონაცემთა ანალიზი. დავუშვათ მოცემული გვაქვს **ASCII** ფაილი altitude.dat, რომელიც შეიცავს იმფორმაციას ორი სვეტის სახით: დრო და ახალი ტიპის მეტეოროლოგიური რაკეტის სიმაღლის შეასაბამისი მნიშვნელობა. ამ მონაცემების საფუძველზე ამოხსენით შემდეგი ამოცანები:

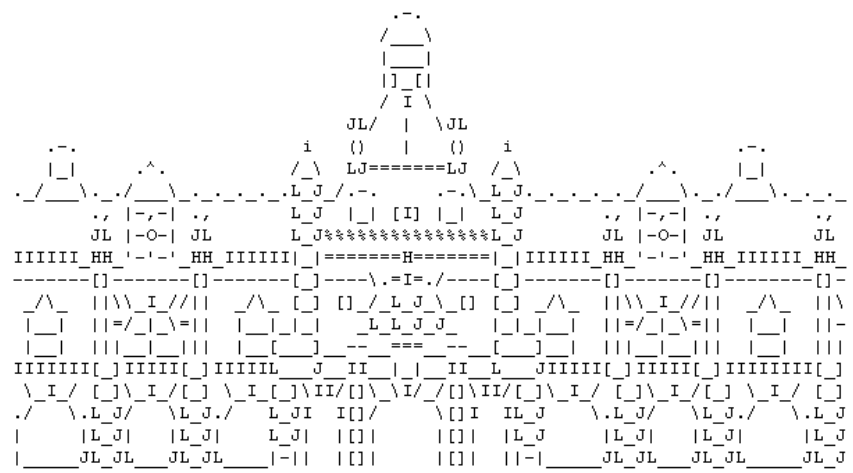
5. გამოთვალეთ და ააგეთ რაკეტის სიჩქარის მნიშვნელობა დროის ყოველ მომენტში მარცხენა დიფერენცირების მეთოდით.
6. გამოთვალეთ და ააგეთ რაკეტის აჩქარების მნიშვნელობა დროის ყოველ მომენტში მარცხენა დიფერენცირების მეთოდით.
7. განსაზღვრეთ რაკეტის საფეხურების, ეტაპების, (stages) რაოდენობა რაკეტისათვის. (მითითება: განიხილე კრიტიკული წერტილები).
8. ააგეთ სიჩქარის გრაფიკი დიფერენცირების სამივე მეთოდის საშუალებით გამოთვლილი მონაცემებისათვის ერთიდაიგივე ნახაზზე.
9. აიღეთ საწყის ინფორმაციად რაკეტის აჩქარების მნიშვნელობები, რომელიც გამოითვალეთ მე-6 ამოცანაში. მათი ინტეგრების საშუალებით მიიღეთ სიჩქარის მნიშვნელობები. (ვერ გამოიყენებთ **quad** ფუნქციას, რადგან მხოლოდ წერტილის (მნიშვნელობები) კოორდინატები გაქვთ. ისარგებლეთ ტრაპეციის ან სიმპსონის წესით.)
10. აიღეთ საწყისად რაკეტის სიჩქარის მნიშვნელობები, რომელიც გამოითვალეთ მე-5 ამოცანაში. მათი ინტეგრების საშუალებით მიიღეთ რაკეტის სიმაღლის შესაბამისი მნიშვნელობები. (ვერ გამოიყენებთ **quad** ფუნქციას, რადგან მხოლოდ წერტილის (მნიშვნელობები) კოორდინატები გაქვთ. ისარგებლეთ ტრაპეციის ან სიმპსონის წესით.)

ფუნქციის ანალიზი. ეს ამოცანა რიცხვით ინტეგრებას უკავშირდება.

11. დავუშვათ მოცემული გვაქვს ფუნქცია:

$$f(x) = 4e^{-x}$$

ააგეთ ფუნქციის გრაფიკი ინტერვალში [0,1]. გამოიყენეთ რიცხვითი ინტეგრების მეთოდი და გამოთვალეთ *f(x)* ფუნქციის ინტეგრალი ინტერვალში[0, 0.5] და [0, 1].



13 სიმბოლური სტრიქონი, სიმბოლური გამონასახის დამუშავება

შესავალი

თუმცა MATLAB –ის ძირითადი დანიშნულება რიცხვითი გამოთვლებია, ზოგჯერ საჭიროა ტექსტური მანიპულაციები. მაგალითად გრაფიკულ გამოსახულებაზე სათაურის გაკეთება, ან ღერძების სახელების დაწერა. ამისათვის MATLAB -ს გააჩნია სტრიქონი, რომელიც ასოით გამოსახულებას შეიცავს, ანუ სიმბოლური სტრიქონი .

13.1 სტრიქონის კონსტრუქცია

MATLAB სიმბოლური სტრიქონი ASCII სიდიდეების ერთობლიობაა. მაგალითად:

```
>>text = ‘This is a character string’
text =
This is a character string
>>size(text)
Ans =
26
>> whos
Name      Size       Bytes       Class
ans         1x2           16           double array
text        1x26          52          char array

Grand total is 28 elements using 68 bytes
```

ASCII კოდი

სტრიქონის ყოველი სიმბოლო წარმოადგენს ერთ ელემენტს მასივში, რომელიც საჭიროებს 2 ბაიტ მეხსიერებას. (რიცხვითი გამოსახულება მოითხოვს 8 ბაიტს.) ASCII კოდი ‘A’- ‘Z’

სიმბოლოთათვის არის ერთმანეთის მომდევნო მთელი რიცხვები 65-დან 90-მდე, ხოლო ‘a’ – ‘z’ 97-დან 122-მდე. MATLAB ფუნქცია **abs** გვაძლევს ASCII კოდს სტრიქონისათვის.

```
>> text = ’This is a character string’
text =
This is a character string
>> d = abs(text)
d =
Columns 1 through 12
84  104  105  115  32  105  115  32  97  32 99  104
Columns 13 through 24
97 114 97 99 16 101 114 32 115 116 114 105
Columns 25 through 26
110  103
```

ფუნქცია **char** ასრულებს შებრუნებულ გარდაქმნას, ASCII კოდის მიხედვით აღადგენს სტრიქონს:

```
>> char(d)
ans =
This is a character string
```

სტრიქონსა და ASCII კოდს შორის ამ დამოკიდებულების საფუძველზე მივიღებთ:

```
>> alpha = abs('a'):abs('z')
>> disp(char(alpha))
abcdefghijklmnopqrstuvwxyz
```

სტრიქონი იგივე მასივია, მაგრამ სიმბოლოთა მასივი, ამიტომ მასზეე შესაძლებელია მასივური ოპერაციების წარმოება:

```
>> text = ‘This is a character string’;
>> u = text(11:19)
u =
character
```

ისევე როგორც მატრიცა, სტრიქონი მასივიც შეიძლება შეიცავდეს მრავალ სტრიქონს, მაგრამ ყოველი სტრიქონი სვეტების თანაბარ რაოდენობას უნდა შეიცავდეს. მაგალითად:

```
v = ['Character strings having more than'
'one row must have the same number '
'of columns - just like matrices   ']
```

მივიღებთ:

```
v =

Character strings having more than
one row must have the same number
of columns - just like matrices

>> size(v)
ans =
```

```
3 34
```

რადგან სტრიქონი იგივე მასივია, შეგვიძლია ორო სტრიქონი გავართიანოთ კვადრატული ფრჩხილების საშუალებით:

```
>> today ='May';
>> today = [today,' 18']
today =
May 18
```

განვიხილოთ MATLAB-ის რამდენიმე ფუნქცია სტრიქონისათვის:

char(x)	x მასივს, რომლის ელემენტებიც მთელი დადებითი რიცხვებია და წარმოადგენს ასოითი სიმბოლოს კოდებს გარდაქმნის ასოით სიმბოლოთა მასივად.
int2str(x)	x მატრიცის ელემენტებს დაამრგვალებს მთელამდე და შედეგად მიღებულ მასივს გარდაქმნის სტრიქონ მატრიცად
num2str(x)	x რიცხვით მატრიცას გარდაქმნის სტრიქონად. ეს ბრძანება სასარგებლოა გრაფიკზე აღნიშვნების გასაკეთებლად (როგორც title, xlabel, ylabel, text ბრძანებათა არგუმენტი)
str2num(s)	s სტრიქონს, რომელიც შეიცავს რიცხვით სიდიდეთა გამოსახულებას ASCII ფორმატში, გარდაქმნის შესაბამის რიცხვით მნიშვნელობად. სტრიქონი s შეიძლება შეიცავდეს ციფრს, ათწილადურ მძიმეს, + ან – ნიშანს, ‘e’ და ‘d’ სიმბოლოებს – მანტისასა და ექსპონეტას გამყოფის სახით და i – კომპლექსური რიცხვებისათვის.

მაგალითები:

```
>> x=abs('string')
x =

   115   116   114   105   110   103

>> char(x)
ans =
string
```

num2str საილუსტრაციოდ განვიხილოთ შემდეგი ბრძანებები:

როგორც ვიცით, ცვლადს - pi MATLAB აღიქვამს როგოც π მნიშვნელობას – 3.1416

```
>> ss=num2str(pi)
ss =
3.1416
```

ss ამ შემთხვევაში არის არა კონკრეტული რიცხვი 3.1416, არამედ სიმბოლოთა მასივი (char array), რომლის ელემენტებიც ASCII კოდებს წარმოადგენს. მართლაც:

```
>> size(ss)
ans =
   1   6
```

```
>> abs(ss)
ans =

  51   46   49   52   49   54
```

ახლა გამოვიყენოთ ბრძანება , რომელიც სტრიქონს ss გარდაქმნის რიცხვით სიდიდედ:

```
pp=str2num(ss)

pp =

    3.1416
```

pp უკვე კონკრეტული რიცხვია (double array)– 3.1416.

```
size(pp)

ans =

1
```

განვიხილოთ კიდევ ერთი მაგალითი:

MATLAB შემდეგ ბრძანებებს:

```
tg = 2.2774;
xg = 144.6364;
disp(['time of flight: ' num2str(tg) ' s'])
disp(['distance traveled : ' num2str(xg) ' ft'])
```

შედეგად მოჰყვება:

```
>> time of flight: 2.2774 s
distance traveled : 144.6364 ft
```

disp ბრძანების პირველი არგუმენტი ჩვეულებრივი სიმბოლური სტრიქონია, მეორე – ბრძანებაა, რომელიც რიცხვით გამოსახულებას სიმბოლურ სტრიქონად გარდაქმნის, და მესამე – ისევ ჩვეულებრივი სიმბოლური სტრიქონი.

MATLAB ფუნქციები სიმბოლოთა სტრიქონისათვის:

blanks(n)	გვაძლევს სტრიქონს , რომელიც n სიმბოლოს ნაცვლად n ცარიელ პოზიციას შეიცავს.
deblank(s)	თუ სტრიქონი ბოლოვდება განმეორებადი ცარიელი პოზიციით, ეს ბრძანება შეკვეცს მას პირველივე სიმბოლომდე
eval(s)	გაუშვებს s სტრიქონს როგორც MATLAB გამოსახულებას ან ბრძანებას.
eval(s1,s2)	საშუალებას იძლევა შეცდომა ვიპოვოთ. გაუშვებს s1 შესაბამის ბრძანებას და გვაძლევს შედგეს, თუ ბრძანება წარმატებით შესრულდა, ხოლო თუ ბრძანების შესრულება შეცდომას იძლევა, უშვებს s2 – შესაბამის ბრძანებას.
findstr(s1,s2)	პოულობს ერთ სტრიქონს მეორის შიგნით
ischar(s)	გვაძლევს 1, თუ s სიმბოლური მასივია, სხვა შემთხვევაში –0

isletter(s)	გვაძლევს 1-ს s სტრიქონის ყოველი ელემენტისათვის, რომელიც ასოით გამოსახულებას წარმოადგენს და 0 სხვა შემთხვევაში.
isspace(s)	გვაძლევს 1-ს s სტრიქონის იმ ელემენტებისათვის, რომლებიც წარმოადგენს ცარიელ პოზიციას(space, tabs, newlines,carriage returns), 0 – სხვა შემთხვევაში

13.2 დრო და თარიღი

უკვე გავეცანით MATLAB ბრძანებას **date**, რომელიც გვაძლევს თარიღს, **clock** – გვაძლევს დროის მნიშვნელობას. უფრო დაწვრილებითი ინფორმაციის მისაღებად ამ საკითხთან დაკავშირებით შეგიძლიათ ისარგებლოთ MATLAB-ის **help** ბრძანებით. გავეცნოთ ზოგიერთ მათგანს.

```
>>[d,s]=weekday('9/15/2006')
```

გვაძლევს კვირის რა დღეა მითითებული თარიღი:

```
d =6
s = Fri

>> calendar('9/15/2006')
```

რომელიც გვაძლევს:

```
                Sep 2006
  S     M     Tu     W     Th     F     S
  0     0      0     0      0     1     2
  3     4      5     6      7     8     9
 10    11     12    13     14    15    16
 17    18     19    20     21    22    23
 24    25     26    27     28    29    30
  0     0      0     0      0     0     0
```

ბრძანება tic ჩართავს ტაიმერს ხოლო toc, გამორთავს მას და მოგვცემს დროს, რომელიც გავიდა ტაიმერის ჩართვის, ანუ tic ბრძანების შემდეგ. ეს ბრძანება სასარგებლოა, როცა გვსურს დავითვალოთ რა დროს ანდომებს კომპიუტერი ჩვენს მიერ დაწერილი პროგრამის ამათუიმ ოპერაციის შესრულებას.

13.3 ბაზური გარდაქმნები და ბიტ ოპერაციები

MATLAB აქვს შესაძლებლობა გადაიყვანოს რიცხვები ორობითიდან ათობითში და ან რომელიმე სხვა ბაზურ სისტემაში სიმბოლური სტრიქონის ფორმით.

dec2bin(d)	გვაძლევს d –ს ორობით წარმოდგენას როგორც სტრიქონს. d უნდა იყოს არაუარყოფითი მთელი რიცხვი $<2^{52}$.
bin2dec(b)	ახდენს b ორობითი სტრიქონის ინტერპრეტირებას და გვაძლევს შესაბამის ათობით რიცხვს

მაგალითად:

```
>> d=32;
>> dec2bin(d)

ans =

100000;

>> bin2dec('100000')

ans =

    32
```

MATLAB –ს გააჩნია ფუნქციები ლოგიკური ოპერაციების განსახორციელებლად რიცხბების შესაბამის ორობით სტრიქონებს შორის. (a, b, c მთელი, არაუარყოფითი რიცხვებია).

c=bitand(a,b)	ბიტური and ორ არგუმენტს შორის
c=bitor(a,b)	ბიტური or a და b არგუმენტს შორის
c=bitcmp(a,N)	a ათობითი მთელი რიცხვის N ბიტიან ორობით წარმოდგენაში შეცვლის 0 1-ით, ხოლო 1 0-ით და მოგვცემს შესაბამის ათობით რიცხვს
c=bitshift(a,N)	მოგვცემს მთელ ათოიბით რიცხვს, რომელიც წარმოადგენს a რიცხვის ორობითი წარმოდგენის N ბიტით წანაცვლების შედეგს. თუ N >0, ეს ოპერაცია იგივეა, რაც გამრავლება 2^N, თუ N < 0, - იგივეა, რაც გაყოფა 2^N-ზე

მაგალითად:

```
>> bitand(13,27)    ბრძანება მოგვცემს:

ans = 9
```

მართლაც რიცხვი 13 ორობითი წარმოდგენა 5 ბიტიან სისტემაში ასეთია :

```
>> dec2bin(13,5)
ans =01101
```

ხოლო - 27-ის:

```
>> dec2bin(27,5)
ans =11011
```

თუ შევადარებთ ერთმანეთს ორ ორობით რიცხვს: 01101 და 11011 **and** შედარების ოპერატორით, მივიღებთ 01001, რასაც შეესაბამება ათობითი რიცვხვი 9. მართლაც:

```
>> bin2dec('01001')
ans = 9
```

ახლა ვნახოთ **bitcmp** ფუნქციის მოქმედების შედეგი:

```
>>bitcmp(9,5) brZanebis Sedegad miviRebT:
ans = 22
```

რადგან ათობითი რიცხვის 9 ორობითი წარმოდგენა 5 ბიტიან სისტემაში = 01001, **bitcmp** ფუნქცია შეცვლის 0 1-ით, ხოლო 1- 0-ით და მიიღება 10110, რასაც შეესაბამება ათობითი რიცხვი 22.

შევამოწმოთ:

```
>> dec2bin(9,5)
ans = 01001

>>dec2bin(22,5)
ans =10110

>>bitShift(9,2)
ans = 36
```

მართლაც 9 ათობითი რიცხვის ორობითი წარმოდგენაა 1001, **bitshift** ბრძანება დაუმატებს მას ორ ნულს მარჯვნივ და მიიღება ორობითი რიცხვი 100100, რაც შეესაბამება ათობით რიცხვს 36, ეს ოპერაცია იგივეა, რაც 9 გავამრავლოთ 2^2 (9*2 2=36).

13.4 სიმბოლური ინფორმაციის დამუშავება

აქმდე ჩვენ MATLAB –ს განვიხილავდით როგორც საშუალებას რიცხვითი გამოთვლების საწარმოებლად. ასევე მოგაწოდეთ გარკვეული ინფორმაცია ტექსტის მანიპულირებისათვის. MATLAB –ის საშუალებით შესაძლებელია ვაწარმოოთ მოქმედებები ჩვეულებრივ ალგებრულ გამოსახულებებზე.

უფრო დაწვრილებითი ინფორმაციისათვის აკრიფეთ ბრძანებათა ფანჯარაში **help symbolic**. ასევე შესაძლებელია ამ საკითხთან დაკავშირებული სადემონსტრაციო მასალის დათვალიერება ბრძანებით **symintro**.

სიმბოლური გამოსახულება და ალგებრა

თუ გვსურს MATLAB-ის საშუალებით სიმბოლური ინფორმაციის დამუშავება, პირველ რიგში უნდა განვმარტოთ (გამოვაცხადოთ) საჭირო მუდმივები და ცვლადები როგორც სიმბოლური ობიექტები. მაგალითად:

```
>>syms x y
```

სიმბოლური მუდმივების გამოსაცხადებლად გამოიყენება ფუნქცია **sym**. მისი არგუმენტია სიმბოლური სტრიქონი, რომელიც შეიცავს სპეციალური ცვლადის სახელს, რიცხვით გამოსახულებას ან ფუნქციის დასახელებას.

```
>> pi = sym('pi');
>> delta = sym('1/10');
>>sqrt2 = sym('sqrt(2) ');
```

თუ სიმბოლური მუდმივი pi ასე განვსაზღვრეთ, იგი მოცემულ სამუშაო სივრცეში შეცვლის pi –ს რიცხვით მნიშვნელობას. სიმბოლური ობიექტების ტიპი განისაზღვრება როგორც ***symbolic objects***.

თუ ზემოხსენებული ბრძანებების შემდეგ მივცემთ ბრძანებას **whos**, მივიღებთ:

```
>> whos
  Name        Size                    Bytes  Class

  delta       1x1                       132  sym object
  pi          1x1                       128  sym object
  sqrt2       1x1                       138  sym object
  x           1x1                       126  sym object
  y           1x1                       126  sym object

Grand total is 20 elements using 650 bytes
```

სიმბოლური გამოსახულება

სიმბოლური ცვლადი შეიძლება გამოვიყენოთ სიმბოლურ გამოსახულებაში როგორც ცვლადი ან ფუნქციის არგუმენტი. არითმეტიკული ოპერატორები + - * / და MATLAB-ის ფუნქციებიც გამოიყენება იგივე წესის მიხედვით, როგორც ეს ხდებოდა რიცხვითი გამოთვლების წარმოებისას. მაგალითად:

```
>> syms s t A
>> f = s^2 + 4*s + 5
f =
s^2+4*s+5
>> g = s + 2
g =
s+2

>> h = f*g
h =
(s^2+4*s+5)*(s+2)
>> z= exp(-s*t)
z =
exp(-s*t)
>> y = A*exp(-s*t)
y =
A*exp(-s*t)
```

პირველ რიგში ვაცხადებთ ცვლადებს t, s და A, შემდეგ ვწერთ სიმბოლურ გამოსახულებებს ახალი ცვლადების f, g, h და y შესაქმნელად. ახლადშექმნილი ცვლადებიც სიმბოლური ცვლადებია და არა რიცხვითი გამოსახულებები.

საზოგადოდ MATLAB-ში თუ არაფერს მივუთითებთ, x არის დამოუკიდებელი ცვლადის სახელი, მაგრამ ეს არ ნიშნავს იმას, რომ სხვა სიმბოლო არ შეგვიძლია გამოვიყენოთ დამოუკიდებელი ცვლადის აღსანიშნავად. დამოუკიდებელი ცვლადის მოსაძებნად სიმბოლურ გამოსახულებაში MATLAB –ს გააჩნია ფუნქცია **findsym**

findsym(S)	S სიმბოლურ გამოსახულებაში ან მატრიცაში პოულობს სიმბოლურ ცვლადებს და გვაძლევს მათ, როგორც სიმბოლურ სტრიქონს, ანბანის მიხედვით დალაგებულს და მძიმეებით

გამოყოფილს. თუ S არ შეიცავს არცერთ სიმბოლურ ცვლადს, გვაძლევს ცარიელ სტრიქონს.

ჩვენს შემთხვევაში :

```
>> findsym(f)
ans =
s
>> findsym(z)
ans =
A, s, t
```

შეგვიძლია შევქმნათ სიმბოლური მატრიცა:

```
>> n = 3;
>> syms x
>> B = x.^((0:n)'*(0:n))

B =

[1,   1,   1,   1]
[1,   x, x^2, x^3]
[1, x^2, x^4, x^6]
[1, x^3, x^6, x^9]
```

როგორ შევასრულოთ სიმბოლური პოლინომის ალგებრული გარდაქმნები:

expand(S)	სიმბოლურ გამოსახულებაში S შეასრულებს ალგებრულ მოქმედებებს, შეკრებს მსგავს წევრებს და წარმოგვიდგენს პოლინომის სახით.
factor(S)	S პოლინომს დაშლის მამრავლებად.
simplify(S)	გაამარტივებს სიმბოლურ გამოსახულებას S
[n,d]=numden(S)	მოგვცემს რაციონალურ წილადად დაშლილი S სიმბოლური გამოსახულების მრიცხველსა და მნიშვნელს
subs(S,old,new)	S სიმბოლურ გამოსახულებაში ცვლადს old შეცვლის ცვლადით new

მაგალითები:

- შეკრება

```
>> syms s
>> A = s^4 -3*s^3 -s +2;
>> B = 4*s^3 -2*s^2 +5*s -16;
>> C = A + B
C =
s^4+s^3+4*s-14-2*s^2
```

- სკალარული გამრავლება

```
>> syms s
>> A = s^4 -3*s^3 -s +2;
>> C = 3*A
C =
3*s^4-9*s^3-3*s+6
```

- გამრავლება

```
>> syms s
>> A = s+2;
>> B = s+3;
>> C = A*B
C =
(s+2)*(s+3)
>> C = expand(C)
C =
s^2+5*s+6
```

- მამრავლებად დაშლა

```
>> syms s
>> D = s^2 + 6*s + 9;
>> D = factor(D)
D =
(s+3)^2
>> P = s^3 - 2*s^2 -3*s + 10;
>> P = factor(P)
P =
(s+2)*(s^2-4*s+5)
```

- გაერთმნიშვნელიანება

განვიხილოთ გამოსახულება

$$H(s) = -\frac{1/6}{s+3} - \frac{1/2}{s+1} + \frac{2/3}{s}$$

სამი შესაკრების გაერთმნიშვნელიანება შესაძლებელია შემდეგნაირად:

```
>> syms s
>> H = -(1/6)/(s+3) -(1/2)/(s+1) +(2/3)/s;
>> [N,D] = numden(H)
N =
s+2
D =
(s+3)*(s+1)*s
>> D = expand(D)
D =
s^3+4*s^2+3*s
```

მაშასადამე შეგვიძლია *H(s)* ასე წარმოვადგინოთ:

$$H(s) = \frac{s+2}{s^3 + 4s^2 + 3s}$$

- შეკვეცა

ვთქვათ გვაქვს სიმბოლური გამოსახულება:

$$H(s) = \frac{s^3 + 2s^2 + 5s + 10}{s^2 + 5}$$

MATLAB მარტივად შეკვეცს ამ წილადს და საბოლოოდ მოგვცემს:

```
>> syms s
>> H = (s^3 +2*s^2 +5*s +10)/(s^2 + 5);
>> H = simplify(H)
H =
s+2
```

თუ წილადის მნიშვნელს დავშლით მამრავლებად, ვნახავთ, რატომ მივიღეთ *s+2*.

```
>> factor(s^3 +2*s^2 +5*s +10)
ans =
(s+2)*(s^2+5)
```

საბოლოოდ გვექნება:

```
H(s) = s + 2
```

- ცვლადის ჩანაცვლება სხვა ცვლადით

განვიხილოთ პოლინომთა თანაფარდობა:

$$H(s) = \frac{s+3}{s^2 + 6s + 8}$$

განვსაზღვროთ მეორე გამოსახულება

$$G(s) = H(s)\,|_{s=s+2}$$

ჩავანაცვლოთ s ცვლადი s+2 –ით MATLAB საშუალებით:

```
>> syms s
>> H = (s+3)/(s^2 +6*s + 8);
>> G = subs(H,s,s+2)
G =
(s+5)/((s+2)^2+6*s+20)
>> G = collect(G)
G =
(s+5)/(s^2+10*s+24)
```

მივიღებთ გამოსახულებას:

$$H(s) = \frac{s+5}{s^2+10s+24}$$

განვიხილოთ კიდევ რამდენიმე სასარგებლო ბრძანება:

sym2poly(P)	P სიმბოლური პოლინომისათვის გვაძლევს ვექტორს, რომლის ელემენტებია ამ პოლინომის კოეფიციენტები
poly2sym(p)	p ვექტორზე დაყრდნობით, რომლის ელემენტებიც პოლინომის კოეფიციენტებს წარმოადგენს, გვაძლევს შესაბამის სიმბოლურ პოლინომს. ცვლადს მიანიჭებს სახელს x.

მაგალითად განვიხილოთ პოლინომი:

$$A(s) = s^3 + 4s^2 - 7s - 10$$

```
>> a = [1 4 -7 -10];
>> A = poly2sym(a,s)
A =
s^3+4*s^2-7*s-10
```

პოლინომისათვის:

$$B(s) = 4s^3 - 2s^2 + 5s - 16$$

```
>> syms s
>> B = 4*s^3 -2*s^2 +5*s -16;
>> b = sym2poly(B)
b =
4   -2    5   -16
```

სიმბოლური გამოსახულების ფორმა

MATLAB-ს შედეგად მიღებული გამოსახულება ყოველთვის არ გამოჰყავს ჩვენთვის სასურველი ფორმით, მაგალითად, გარკვეული ოპერაციების შესრულების შემდეგ შედეგად შესაძლოა მივიღოთ 1/a*b, და არა b/a.

ტრიგონომეტრიული სიმბოლური გამოსახულება

განვიხილოთ შემდეგი ბრძანებები:

```
>> syms theta phi
>> A = sin(theta + phi)
A =
sin(theta+phi)
>> A = expand(A)
A =
sin(theta)*cos(phi)+cos(theta)*sin(phi)
>> B = cos(2*theta)
B =
cos(2*theta)
```

```
>> B = expand(B)
B =
2*cos(theta)^2-1
>> C = 6*((sin(theta))^2 + (cos(theta))^2)
C =
6*sin(theta)^2+6*cos(theta)^2
>> C = expand(C)
C =
6*sin(theta)^2+6*cos(theta)^2
```

როგორც ვნახეთ MATLAB –მა "კარგად გაართვა თავი" რთულ ტრიგონომეტრიულ გარდაქმნებს, თუმცა ბოლო სრულიად მარტივი გარდაქმნის შესრულება "გაუჭირდა":

$$C = 6(\sin^2(\theta) + \cos^2(\theta)) = 6$$

სიმბოლური გამოსახულების რიცხვითი მნიშვნელობის გამოთვლა და გრაფიკის აგება

ხშირად გვჭირდება ცვლადის მოცემული მნიშვნელობისათვის გამოვთვალოთ სიმბოლური გამოსახულების მნიშვნელობა. ამისათვის საჭიროა ფუნქციით **subs** სიმბოლურ ცვლადს მივანიჭოთ რიცხვითი მნიშველობა:

```
>> E = s^3 -14*s^2 +65*s -100;
>> F = subs(E,s,7.1)

F =

   13.6710
```

F პირდაპირ ღებულობს რიცხვით მნიშვნელობას double array

შესაძლებელია სიმბოლური გამოსახულების მიხედვით ავაგოთ გრაფიკი:

ezplot(f)	ააგებს f(x) გრაფიკს, სადაც f სიმბოლური გამოსახულებაა და წარმოადგენს მათემატიკურ გამოსახულებას, რომელიც შეიცავს მხოლოდ ერთ ცვლადს, ვთქვათ x. მითითების გარეშე (default) x მნიშვნელობათა არეა [-2π, 2π]
ezplot(f,xmin,xmax)	ააგებს გრაფიკს x მითითებულ მნიშვნელობათა მიხედვით

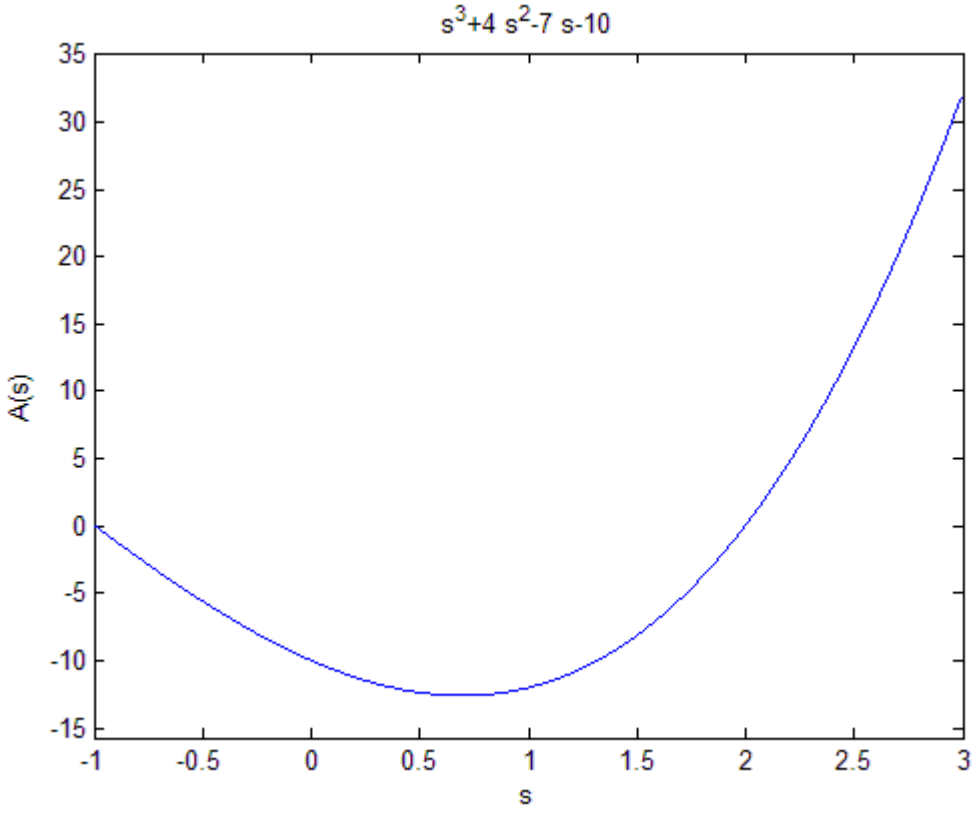
საილუსტრაციოდ განვიხილოთ ფუნქცია:

$$A(s) = s^3 + 4s^2 - 7s - 10$$

ინტერვალში [-1, 3]:

```
syms s
a = [1 4 -7 -10];
A = poly2sym(a,s)
ezplot(A,-1,3), ylabel('A(s)')
```

შედეგად მიღებული გრაფიკი ნაჩვენებია ნახ. 13.1.



ნახ. 12.1 ezplot ფუნქციის საშუალებით აგებული პოლინომის გრაფიკი

მიაქციეთ ყურადღება, რომ გამოსახულება, რომლის მიხედვითაც აიგო გრაფიკი ავტომატურად განთავსდება გრაფიკის სათაურად, ხოლო x ღერძს ავტომატურად წაეწერება ცვლადის სახელი (ამ შემთხვევაში s)

სიმბოლური ალგებრული და ტრანსცენდენტული განტოლებები

MATLAB –ის საშუალებით შესაძლებელია ამოვხსნათ სიმბოლური ალგებრული ან ტრანსცენდენტული განტოლებები და განტოლებათა სისტემები. ტრანსცენდენტული ისეთ განტოლებას ეწოდება, რომელიც შეიცავს რომელიმე ტრანსცენდენტულ ფუნქციას: *cos x*, e^x ან *ln x*. ამისათვის გამოიყენება ფუნქცია solve. არსებობს ამ ფუნქციის რამდენიმე ფორმა, ჩვენ განვიხილავთ შემდეგს:

```
solve(E1, E2, E3,. . .,EN)
solve(E1, E2, E3,. . .,EN, var1, var2, . . ., var3)
```

სადაც E1, E2, E3,. . ., სიმბოლურ გამოსახულებათა სახელებია, ხოლო var1, var2, . . ., var3 შესაბამისი ცვლადები. მიღებული ამონახსნები წარმოადგენს გამოსახულების ფესვებს. ეს არის სიმბოლური გამოსახულებანი ცვლადებისათვის შემდეგი პირობებით E1=0, E2=0, E3=0,. . .,EN=0.

ერთი განტოლებისათვის ერთი ცვლადით ამონახსნი იქნება ერთი სიმბოლური სიდიდე. მაგალითად:

```
>> syms s
>> E = s+2;
>> s = solve(E)
s =
-2
```

მივიღეთ სიმბოლური სიდიდე s=-2. იმისათვის, რომ იგი რიცხვით გამოსახულებად გარდავქმნათ საჭიროა მივმართოთ ბრძანებას **double:**

double(S)	სიმბოლურ გამოსახულებას S გარდაქმნის რიცხვით სიდიდედ (double prescision floating point number). S არ უნდა შეიცავდეს სიმბოლურ ცვლადს.

```
>> snum=double(s)

snum =

    -2
```

განვიხილოთ სიმბოლური გამოსახულება – მეორე ხარისხის პოლინომი:

```
>> syms s
>> D = s^2 +6*s +9;
>> s = solve(D)
s =
[ -3]
[ -3]
```

მაგალითი ტრიგონომერტიიდან:

```
>> syms theta x z
>> E = z*cos(theta) - x;
>> theta = solve(E,theta)
theta =
acos(x/z)
```

განტოლებებს, რომლებიც შეიცავენ პერიოდულ ფუნქციებს, შესაძლოა ჰქონდეთ ფესვების განუსაზღვრელი რაოდენობა. ასეთ შემთხვევაში **solve** ფუნქცია შემოიფარგლება ფესვებით 0-ის მახლობლობაში. მაგალითად ამოვხსნათ განტოლება:

$$2\cos(2\theta) - \sin(\theta) = 0$$

```
>> E = cos(2*theta)-sin(theta);
>> solve(E)
ans =
[ -1/2*pi]
[ 1/6*pi]
[ 5/6*pi]
```

MATLAB –ში შესაძლებელია სიმბოლური გამოსახულების ინტეგრება და დიფერენცირება.

diff(E)	ახდენს E სიმბოლური გამოსახულების დიფერენცირებას იმ ცვლადით, რომელიც განისაზღვრება ფუნქციით findsym(E)
diff(E,v)	ახდენს E სიმბოლური გამოსახულების დიფერენცირებას v ცვლადით
diff(E,n)	ახდენს E სიმბოლური გამოსახულების დიფერენცირებას n-ჯერ(n მთელი დადებითი რიცხვია)
diff(S,v,n)	ახდენს E სიმბოლური გამოსახულების დიფერენცირებას v ცვლადით n-ჯერ(n მთელი დადებითი რიცხვია)

პოლინომის წარმოებულის გამოთვლა:

```
>> syms s n
>> p = s^3 + 4*s^2 -7*s -10;
>> d = diff(p)
d =
3*s^2+8*s-7
>> e = diff(p,2)
e =
6*s+8
>> f = diff(p,3)
f =
6
>> g = s^n;
>> h = diff(g)
h =
s^n*n/s
>> h = simplify(h)
h =
s^(n-1)*n
```

ტრანსცენდენტული ფუნქციის წარმოებული:

```
>> syms x
>> f1 = log(x);
>> df1 = diff(f1)
df1 =
1/x
>> f2 = (cos(x))^2;
>> df2 = diff(f2)
df2 =
-2*cos(x)*sin(x)
>> f3 = sin(x^2);
>> df3 = diff(f3)
df3 =
2*cos(x^2)*x
>> df3 = simplify(df3)
df3 =
2*cos(x^2)*x
>> f4 = cos(2*x);
>> df4 = diff(f4)
df4 =
-2*sin(2*x)
```

```
>> f5 = exp(-(x^2)/2);
>> df5 = diff(f5)
df5 =
-x*exp(-1/2*x^2)
```

ინტეგრება

int(E)	გვაძლევს სიმბოლური გამოსახულების განუსაზღვრელ ინტეგრალს მისი სიმბოლური ცვლადის მიხედვით, რომელიც განისაზღვრება ფუნქციით findsym
int(E,v)	E სიმბოლური გამოსახულების განუსაზღვრელი ინტეგრალი v ცვლადის მიხედვით
int(E,a,b)	E სიმბოლური გამოსახულების განსაზღვრული ინტეგრალი ინტერვალში [a b], სადაც a და b სიმბოლური ან რიცხვითი სკალარებია
int(E,v,a,b)	E სიმბოლური გამოსახულების განსაზღვრული ინტეგრალი v ცვლადის მიხედვით ინტერვალში [a b], სადაც a და b სიმბოლური ან რიცხვითი სკალარებია

პოლინომის ინტეგრება:

```
>> syms x n a b t
>> int(x^n)
ans =
x^(n+1)/(n+1)
>> int(x^3 +4*x^2 + 7*x + 10)
ans =
1/4*x^4+4/3*x^3+7/2*x^2+10*x
>> int(x,1,t)
ans =
1/2*t^2-1/2
>> int(x^3,a,b)
ans =
1/4*b^4-1/4*a^4
```

ტრანსცენდენტული ფუნქციის ინტეგრება:

```
>> syms x
>> int(1/x)
ans =
log(x)
>> int(cos(x))
ans =
sin(x)
>> int(1/(1+x^2))
ans =
atan(x)
>> int(exp(-x^2))
ans =
1/2*pi^(1/2)*erf(x)
```

erf(x) წარმოადგენს ცთომილების ინტეგრალს (error function) ყოველი x-თვის, სადაც x – ნამდვილი რიცხვია, ხოლო erf(x) შემდეგნაირად განისაძღვრება:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

წრფივი ალგებრა

მატრიცული ოპერაციების განხორციელება შესაძლებელია სიმბოლური მატრიცებისთვისაც.

```
>> A = sym([2,1; 4,3])
A =
[ 2, 1]
[ 4, 3]
>> Ainv = inv(A)
Ainv =
[ 3/2, -1/2]
[ -2, 1]
>> C = A*Ainv
C =
[ 1, 0]
[ 0, 1]
>> B = sym([1 3 0; -1 5 2; 1 2 1])
B =
[ 1, 3, 0]
[ -1, 5, 2]
[ 1, 2, 1]
>> detB = det(B)
detB =
10
```

ასევე შესაძლებელია MATLAB საშუალებით ამოვხსნათ წრფივ განტოლებათა სიმბოლური სისტემა:

```
>> syms x
>> A = sym([3 2 -1; -1 3 2; 1 -1 -1])
A =
[ 3, 2, -1]
[ -1, 3, 2]
[ 1, -1, -1]
>> b = sym([10; 5; -1])
b =
[ 10]
[ 5]
[ -1]
>> x = A\b
x =
[ -2]
[ 5]
[ -6]
```

მიღებული შედეგი შეადარეთ იგივე სისტემის რიცხვით ამონახსნებს, რათა დარწმუნდეთ პასუხის სისწორეში. თუმცა, განტოლებათა სისტემის ამოხსნისას რაიმე პარამეტრის მიმართ, მცირედი უპირატესობა გააჩნია სიმბოლურ მიდგომას. განვიხილოთ განტოლებათა სისტემა:

$$2x_1 - 3x_2 = 3$$
$$5x_1 + cx_2 = 19$$

ამოვხსნათ x_1 და x_2 როგორც c პარამეტრის ფუნქცია:

```
>> syms c
>> A = sym([2 -3; 5 c]);
>> b = sym([3; 19]);
>> x = A\b
x =
[ 3*(19+c)/(2*c+15)]
[ 23/(2*c+15)]
```

ბრძანების **subs** საშუალებით შეგვიძლია ვიპოვოთ განტოლებათა სისტემის ამონახსნები c სხვადასხვა მნიშვნელობათათვის:

თუ c=1, მივიღებთ:

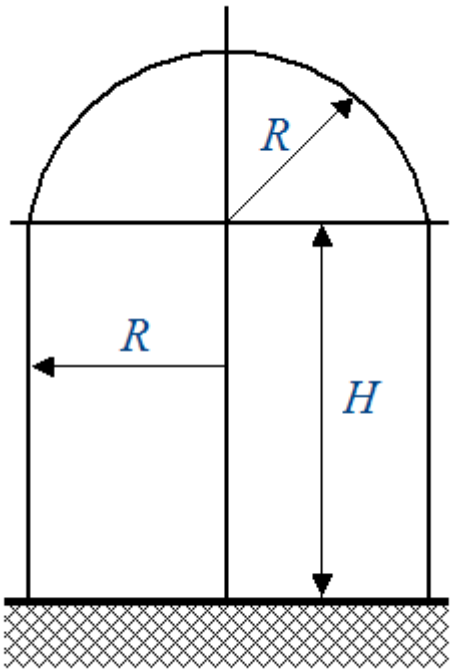
```
x1=subs(x,1)

x1 =

    3.5294
    1.3529
```

ცილინდრული ფორმის რეზერვუარის დიზაინი მინიმალური დანახარჯით

უნდა აიგოს ცილინდრული ფორმის რეზერვუარი ნახევარსფეროს ფორმის გადახურვით მინიმალური დანახარჯით, თუ ცნობილია, რომ ცილინდრული ზედაპირის კონსტრუქცია ღირს \$300/მ^2, ხოლო ცფერული ზედაპირის - \$400/მ^2, ხოლო რეზერვუარის ტევადობაა $5*10^5$ ლ. რეზერვუარი უნდა აიგოს ბეტონის ზედაპირზე, რომლის ფასიც მხედველობაში არ უნდა იქნას მიღებული.

1. ამოცანის დასმა

ნახ. 12.2 –ზე მოცემულია რეზერვუარის ზომები. განვსაზღვროთ ცილინდრის რადიუსი და სიმაღლე ისე, რომ რეზერვუარის ასაგებად მინიმალური თანხა დაიხარჯოს



ნახ. 12.2 რეზერვუარის კონსტრუქცია

2. INPUT/OUTPUT აღწერა

ცნობილია, რომ ცილინდრული ზედაპირის აგება ღირს \$300/მ^2, ხოლო ცფერული ზედაპირის - \$400/მ^2, ხოლო ასაგები რეზერვუარის ტევადობაა $5*10^5$ ლ. უნდა განვსაზღვროთ ცილინდრის რადიუსი და სიმაღლე ისე, რომ რეზერვუარის აგებაზე მინიმალური თანხა დაიხარჯოს.

3. სახელდახელო ამოხსნა

ცილინდრის მოცულობა: $V_c = \pi \cdot R^2 \cdot H$

ნახევარსფეროს მოცულობა: $V_h = \frac{2}{3}\pi \cdot R^3$

ცილინდრის ზედაპირის ფართობი: $A_c = 2\pi \cdot R \cdot H$

ნახევარსფეროს ზედაპირის ფართობი: $A_h = 2\pi \cdot R^2$

რადგან რეზერვუარიის მოცულობაა 500ლ = 500 მ^3, შეგვიძლია დავწეროთ გამოსახულება:

$$500 = \pi \cdot R^2 \cdot H + \frac{2}{3}\pi \cdot R^3$$

აქედან

$$H = \frac{500}{\pi R^2} - \frac{2R}{3}$$

გამოვსახოთ რეზერვუარის ღირებულება როგორც ცილინდრის სიმაღლისა და რადიუსის ფუნქცია:

$$C = 300 \cdot A_c + 400 \cdot A_h = 300 \cdot (2 \cdot \pi R \cdot H) + 400(2 \cdot \pi \cdot R^2)$$

როგორც ვიცით, უწყვეტი ფუნქციის წარმოებული ნულს უტოლდება ფუნქციის მინიმუმისა და მაქსიმუმის წერტილებში. პირველ რიგში მოცულობის გამოსათვლელი განტოლებიდან

ვიპოვოთ H როგორც R-ის ფუნქცია. შემდეგ გავაწარმოოთ C ფუნქცია R-ით, გავუტოლოთ მიღებული გამოსახულება 0-ს და ვიპოვოთ განტოლების ფესვები. ასე ვიპოვით რადიუსის მნიშვნელობას, რომელიც შეესაბამება მინიმალურ ღირებულებას, ხოლო რადიუსის საშუალებით გამოვითვლით სიმაღლის სათანადო მნიშვნელობასაც.

4. MATLAB ამოხსნა

```
syms R H
V = pi*R^2*H + (2/3)*pi*R^3 -500; % Equation for volume
H = solve(V,H);                   % Solve volume for height H
H =-2/3*(pi*R^3-750)/pi/R^2;
C = 300*(2*pi*R*H) + 400*(2*pi    % Equation for cost
dCdR = diff(C,R);                 % Derivative of cost wrt R
dCdR =400/R^2*(pi*R^3-750)+400*pi*R;
Rmins = solve(dCdR,R);            % Solve dC/dR for R: Rmin
Rmins = double(Rmins);

Rmin = Rmins(1);
Hmin = double(subs(H,R,Rmin));
Cmin = double(subs(C,{R,H},{Rmin,Hmin}));
```

5. შემოწმება

პროგრამა გამოითვლის რეზერვუარის ასაგებად საჭირო მინიმალურ დანახარჯს და რადიუსის და სიმაღლის სათანადო მნიშვნელობებს.

```
Rmin =
   4.92372510921348
Hmin =

   3.28248340614232
Cmin =
   9.139421678069333e+004
```

ბრძანებები და ფუნქციები

char(x)	x მასივს, რომლის ელემენტებიც მთელი დადებითი რიცხვებია და წარმოადგენს სიმბოლოთა შესაბამის კოდებს, გარდაქმნის სიმბოლოთა სტრიქონად.
[n,d]=numden(S)	მოგვცემს რაციონალურ წილადად დაშლილი S სიმბოლური გამოსახულების მრიცხველსა და მნიშვნელს
bin2dec(b)	ახდენს ორობითი რიცხვის - b ინტერპრეტირებას და გვაძლევს შესაბამის ათობით რიცხვს
blancs(n)	გვაძლევს სტრიქონს , რომელიც შეიცავს n სიმბოლოს ნაცვლად n ცარიელ პოზიციას.
c=bitand(a,b)	ბიტური and ორ არგუმენტს შორის
c=bitcmp(a,N)	a ათობითი მთელი რიცხვის N ბიტიან ორობით წარმოდგენაში შეცვლის 0 1-ით, ხოლო 1 0-ით და მოგვცემს შესაბამის ათობით რიცხვს
c=bitor(a,b)	ბიტური or a და b არგუმენტს შორის
c=bitshift(a,N)	მოგვცემს მთელ ათოიბით რიცხვს, რომელიც წარმოადგენს a რიცხვის ორობითი წარმოდგენის N ბიტით წანაცვლების შედეგს. თუ N >0, ეს ოპერაცია იგივეა, რაც გამრავლება 2^N,

	თუ $N < 0$, - იგივეა, რაც გაყოფა 2^N-ზე
deblank(s)	თუ სტრიქონი ბოლოვდება განმეორებადი ცარიელი პოზიციით, ეს ბრძანება შეკვეცს მას პირველივე სიმბოლომდე
dec2bin(d)	გვაძლევს d ორობით წარმოდგენას როგორც სტრიქონს. d უნდა იყოს არაუარყოფითი მთელი რიცხვი.
dec2hex(d)	გვაძლევს d თექვსმეტობით წარმოდგენას როგორც სტრიქონს. d უნდა იყოს არაუარყოფითი მთელი რიცხვი.
diff(E)	ახდენს E სიმბოლური გამოსახულების დიფერენცირებას იმ ცვლადის მოხედვით, რომეიც განისაზღვრება ფუნქციით findsym(E)
diff(E,n)	ახდენს E სიმბოლური გამოსახულების დიფერენცირებას n-ჯერ(n მთელი დადებითი რიცხვია)
diff(E,v)	ახდენს E სიმბოლური გამოსახულების დიფერენცირებას v ცვლადის მიხედვით
diff(S,v,n)	ახდენს E სიმბოლური გამოსახულების დიფერენცირებას v ცვლადის მიხედვით n-ჯერ(n მთელი დადებითი რიცხვია)
double(S)	სიმბოლურ გამოსახულებას S გარდაქმნის რიცხვით სიდიდედ (double prescision floating point number). S არ უნდა შეიცავდეს სიმბოლურ ცვლადს.
eval(s)	გაუშვებს s სტრიქონს როგორც MATLAB გამოსახულებას ან ბრძანებას.
eval(s1,s2)	საშუალებას იძლევა შეცდომა დავიჭიროთ. გაუშვებს s1 შესაბამის ბრძანებას და გვაძლევს შედგეს, თუ ბრძანება წარმატებით შესრულდა, ხოლო თუ ბრძანების შესრულება შეცდომას იძლევა, უშვებს s2 – შესაბამის ბრძანებას.
expand(S)	სიმბოლურ გამოსახულებაში S შეასრულებს ალგებრულ მოქმედებებს, შეკრებს მსგავს წევრებს და წარმოგვიდგენს პოლინომის სახით.
ezplot(f)	ააგებს f(x) გრაფიკს, სადაც f სიმბოლური გამოსახულებაა და წარმოადგენს მათემატიკურ გამოსახულებას, რომელიც შეიცავს მხოლოდ ერთ ცვლადს, ვთქვათ x. მითითების გარეშე (default) x მნიშვნელობათა არეა $[-2\pi, 2\pi]$
ezplot(f,xmin,xmax)	ააგებს გრაფიკს x მითითებულ ინტერვალში
factor(S)	S პოლინომს დაშლის მამრავლებად.
findstr(s1,s2)	პოულობს ერთ სტრიქონს მეორის შიგნით
findsym(S)	სიმბოლურ გამოსახულებაში ან მატრიცაში S პოულობს სიმბოლურ ცვლადებს და გვაძლევს მათ, როგორც სიმბოლურ სტრიქონს, ანბანის მიხედვით დალაგებულს და მძიმეებით გამოყოფილს. თუ S არ შეიცავს არცერთ სიმბოლურ ცვლადს, გვაძლევს ცარიელ სტრიქონს.
hex2dec(h)	ახდენს ორობითი რიცხვის h ინტერპრეტირებას და გვაძლევს შესაბამის ათობით რიცხვს
int(E)	გვაძლევს სიმბოლური გამოსახულების განუსაზღვრელ ინტეგრალს მისი სიმბოლური ცვლადის მიხედვით, რომელიც განისაზღვრება ფუნქციით findsym
int(E,a,b)	E სიმბოლური გამოსახულების განსაზღვრული ინტეგრალი ინტერვალში [a b], სადაც a და b სიმბოლური ან რიცხვითი სკალარებია

int(E,v)	E სიმბოლური გამოსახულების განუსაზღვრელი ინტეგრალი v ცვლადის მიხედვით
int(E,v,a,b)	E სიმბოლური გამოსახულების განსაზღვრული ინტეგრალი v ცვლადის მიხედვით ინტერვალში [a b], სადაც a და b სიმბოლური ან რიცხვითი სკალარებია
int2str(x)	x მატრიცის ელემენტებს დაამრგვალებს მთელამდე და შედეგად მიღებულ მასივს გარდაქმნის სიმბოლურ სტრიქონად
ischar(s)	გვაძლევს 1, თუ s სიმბოლური მასივია, სხვა შემთხვევაში –0
isletter(s)	გვაძლევს 1 s სტრიქონის ყოველი ელემენტისათვის, რომელიც ასოით გამოსახულებას წარმოადგენს და 0 სხვა შემთხვევაში.
isspace(s)	გვაძლევს 1 s სტრიქონის იმ ელემენტებისათვის, რომლებიც წარმოადგენს ცარიელ პოზიციას(space, tabs, newlines,carriage returns), 0 – სხვა შემთხვევაში
num2str(x)	x მატრიცას გარდაქმნის სტრიქონად. ეს ბრძანება სასარგებლოა გრაფიკზე აღნიშვნების გასაკეთებლად (title, xlabel, ylabel, text)
poly2sym(p)	p ვექტორზე დაყრდნობით, რომლის ელემენტებიც პოლინომის კოეფიციენტებს წარმოადგენს, გვაძლევს შესაბამის სიმბოლურ პოლინომს. ცვლადს მიანიჭებს სახელს x.
simplify(S)	გაამარტივებს სიმბოლურ გამოსახულებას S
str2num(s)	გარდაქმნის s სტრიქონს, რომელიც უნდა იყოს რიცხვითი სიდიდის შესაბამისი ASCII კოდი, შესაბამის რიცხვით მნიშვნელობად. სტრიქონი s შეიძლება შეიცავდეს ციფრს, ათწილადურ მძიმეს, + ან – ნიშანს, ‘e’ 10-ის ხარისხის მაჩვენებლს და I – კომპლექსური რიცხვებისათვის.
subs(S,old,new)	S სიმბოლურ გამოსახულებაში ცვლადს old შეცვლის ცვლადით new
sym2poly(P)	სიმბოლური პოლინომისათვის P გვაძლევს ვექტორს, რომლის ელემენტებია ამ პოლინომის კოეფიციენტები

14 სიგნალის დამუშავება

მანქანური ხედვა მოითხოვს ციფრული გამონასახის სწრაფ დამუშავებასა და გაიგივებას. ეს პროცესი საჭიროებს არა მხოლოდ სწრაფ ალგორითმს, არამედ დახვეწილ და ფაქიზ ტექნოლოგიას, რომელიც უზრუნველყოფს დიდი მოცულობის გამონასახის სწრაფ ჩამოტვირთვას კომპიუტერის მეხსიერებაში. გამონასახის დამუშავების პროცესი მოიცავს რამდენიმე საფეხურს, რომელიც სასურველია განხორციელდეს ერთმანეთის პარალელურად. ამ პროცესში მნიშვნელოვანი ადგილი უკავია ციფრული ფილტრაციის ოპერაციას. ამ ამოცანას ემსახურება ციფრული მიკროპროცესორი, რომელიც წარმოადგენს ციფრულ ელექტრონულ კომპონენტს ნახევარგამტარულ ელექტრონულ სქმაზე დამონტაჟებული ტრანზისტორებით. კომპიუტერის ცენტრალური პროცესორი ერთი ან რამდენიმე მიკროპროცესორისაგან შედგება.

14.1 სიხშირული ანალიზი

ამ თავში განვიხილავთ MATLAB –ის ფუნქციებს, რომლებიც დაკავშირებულია სიგნალის დამუშავებასთან. განვიხილავთ ოთხი კატეგორიის ფუნქციებს: სიხშირული ანალიზი, ფილტრის ანალიზი, ფილტრის დიზაინი და ფილტრის მოდელი. განვსაზღვრავთ აღნიშვნებს, რომლებიც გამოიყენება სიგნალის დამუშავების თეორიაში. განვიხილავთ როგორც ანალოგური, ასევე ციფრული სიგნალის დამუშავების თეორიას. უფრო დაწვრილებით – ციფრულ სიგნალს.

ანალოგური სიგნალი არის უწყვეტი ფუნქცია(ჩვეულებრივ დროისა) რომელიც წარმოადგენს გარკვეულ ინფორმაციას, მაგალითად: ხმოვანი სიგნალი, სისხლის წნევის გამომხატველი სიგნალი, ანდა სეისმური სიგნალი. იმისათვის, რომ ეს ინფორმაცია დავამუშაოთ კომპიუტერის საშუალებით, ანალოგური სიგნალი უნდა აღირიცხოს ყოველ T წამში და ასე შეიქმნას ციფრული სიგნალი, რომელიც წარმოადგენს საწყისი ანალოგური სიგნალის რიცხვითი სიდიდეების მწკრივს.

$$f_k = f(kT)$$

ციფრული სიგნალი არის შერჩეულ სიდიდეთა მწკრივი f_k.

დრო, როცა დავიწყეთ ანათვლების აღება, ჩვეულებრივ ითვლება ნულის ტოლად და პირველი მნიშვნელობაც მწკრივში იქნება f_0.

თუ ანათვლებს ვიღებთ 100 ჰც სიხშირით (წამში 100 ანათვალი), ციფრული სიგნალის პირველი 3 მნიშვნელობა იქნება:

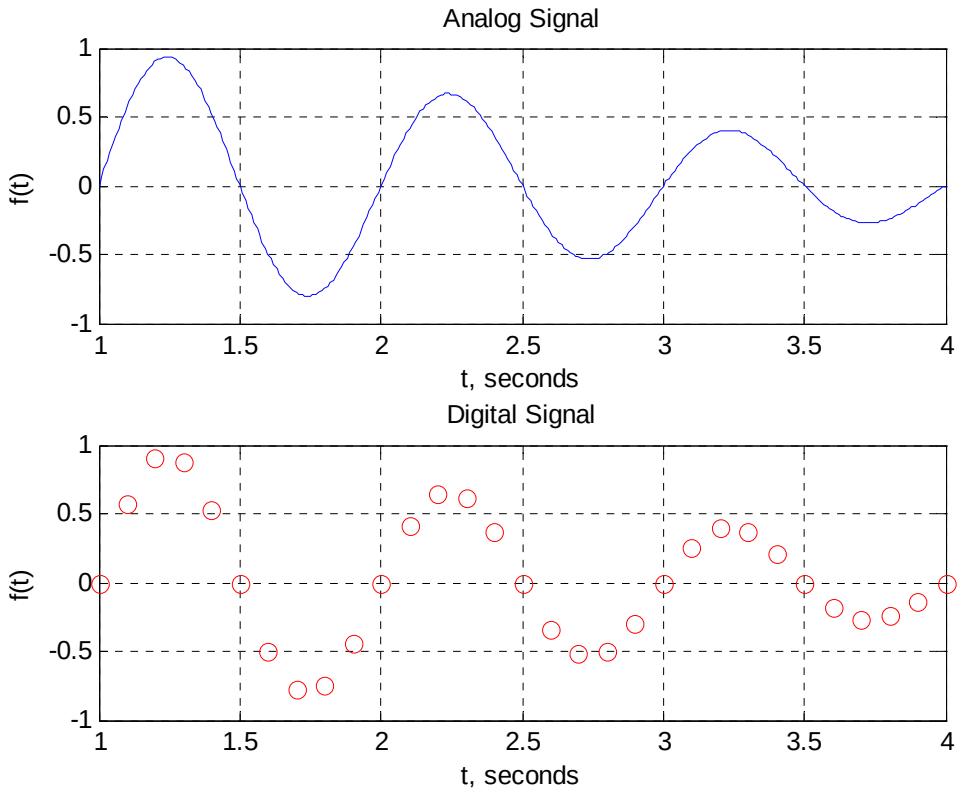
$$f_0 = f(0T) = f(0.0)$$
$$f_1 = f(1T) = f(0.01)$$
$$f_2 = f(2T) = f(0.02)$$

სიგნალის ანალიზი წარმოებს დროით და სიხშირულ გარემოში. სიხშირული სიგნალი შესაძლოა წარმოდგენილი იყოს კომპლექსური რიცხვების სახით, როგორც ჯამი სინუსოიდებისა (კოსინუსური ფორმით), რომელთაგანაც შედგება სიგნალი (ჰარმონიკების სახით). ზოგიერთი ინფორმაცია უკეთ ჩანს სიგნალის დროითი ანალიზიდან, მაგალითად თუ შევხედავთ დროით მრუდს, შეგვიძლია ვთქვათ პერიოდულია თუ არა სიგნალი. დროითი მონაცემებით გამოვითვლით სიგნალის საშუალოს, სტანდარტულ გადახრას, გადახრას და სიმძლავრეს. სიგნალის სიხშირული მახასიათებლების შესაფასებლად კი გვჭირდება მისი სიხშირული ანალიზი. სიგნალის სიხშირულ შემცველობას მის სპექტრს უწოდებენ.

დროითი სიგნალის გადასაყვანად სიხშირულ გარემოში გამოიყენება ფურიეს დისკრეტული გარდაქმნა. საწყისი მონაცემები ფურიეს გარდაქმნისათვის არის დროითი მონაცემების მწკრივი f_k. ალგორითმი გამოითვლის კომპლექსურ სიდიდეთა მწკრივს F_k რომელიც წარმოადგენს სიხშირულ ინფორმაციას. **DFT** ალგორითმი მოითხოვს საკმაო დროს, როცა მონაცემთა მწკრივი გრძელია, მაგრამ თუ მონაცემთა რაოდენობა 2-ის ხარისხის ტოლია ($N = 2^m$), გამოიყენება განსხვავებული ალგორითმი - ფურიეს სწრაფი გარდაქმნა (**fft**), რომელიც საგრძნობლად ამცირებს გამოთვლით დროს.

რადგან ციფრული სიგნალი აითვლება ყოველ T წამში, წამში გვექნება 1/T ანათვალი ანუ ათვლის სიხშირე იქნება 1/T ჰერცი. ათვლის სიხშირე სიფრთხილით უნდა შეირჩეს, რათა თავიდან ავიცილოთ ზედდების ეფექტი (aliasing) – პრობლემა, რომელიც გამოწვეულია იმით, რომ შერჩევა არ ხდება შესაბამისი სიხშირით. შეიძლება ვაჩვენოთ, რომ ამის თავიდან ასაცილებლად სიგნალის ათვლის სიხშირე უნდა აღემატებოდეს სიგნალის შემადგენელი

ნებისმიერი სინუსოიდის სიხშირის გაორმაგებულ მნიშვნელობას. მაგალითად თუ სიგნალი შეიცავს ორ სინუსოიდას, ერთის სიხშირეა 10 ჰც, მეორის კი 35 ჰც, სიგნალის ათვლის სიხშირე უნდა იყოს 70 ჰც. **ნაიქვისტის სიხშირე** ტოლია ათვლის სიხშირე/2 და წარმოადგენს ზედა საზღვარს იმ სიხშირეებისას, რომელსაც შეიძლება შეიცავდეს ციფრული სიგნალი.



ნახ. 14.1 ანალოგური და ციფრული სიგნალი

MATLAB –ის ფუნქცია ***fft*** გამოითვლის სიგნალის სიხშირულ სპექტრს. მას შეიძლება ჰქონდეს ერთი ან ორი არგუმენტი. თუ მხოლოდ ერთი არგუმენტი აქვს, ეს უნდა იყოს ვექტორი, რომლის ელემენტებია სიგნალის ამპლიტუდები დროის განსაზღვრულ ინტერვალში. შედეგად უნდა მივიღოთ იგივე სიგრძის ვექტორი, რომელიც წარმოადგენს კომპლექსურ რიცხვთა მწკრივს და გვაძლევს საწყისი სიგნალის სიხშირულ სპექტრს.

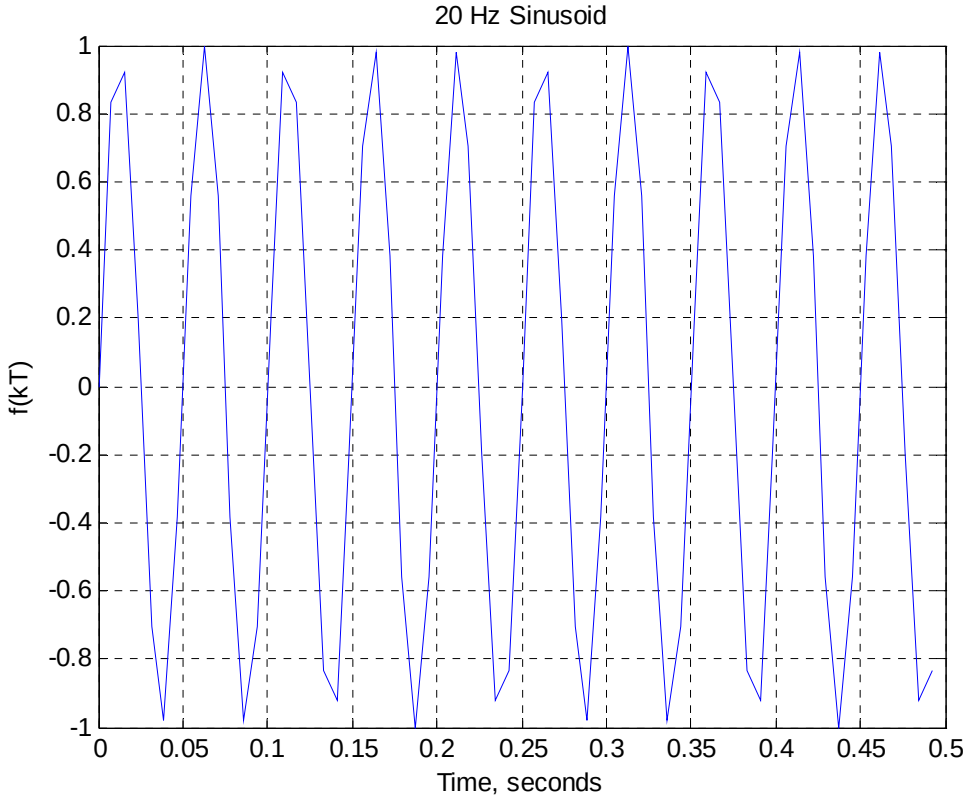
fft შედეგად მიღებული ვექტორის ურთიერთმომდევნო ელემენტები ერთმანეთისგან განსხვავდებიან 1/NT ჰერცით. ე. ი. თუ გვაქვს დროითი სიგნალის 32 ანათვალი, რომელიც ათვლილი იყო 1000 ჰც სიხშირით, ***fft*** ალგორითმით დათვლილი მნიშვნელობები შეესაბამება 0ჰც, 1/0.032ჰც, 2/0.032ჰც და ა.შ. ანუ 0ჰც, 31.25ჰც, 62.5ჰც ... ნაიქვისტის სიხშირე = 1/2T და ეს შეესაბამება F_{16} . რადგან ფურიეს დისკრეტული ფუნქცია პერიოდულია, ნაიქვისტის სიხშირის ზევით მონაცემები ახალ ინფორმაციას არ შეიცავს და ვსარგებლობთ შედეგად მიღებული ვექტორის ელემენტთა მხოლოდ პირველი ნახევრით.

განვიხილოთ MATLAB-ის ბრძანებათა შემდეგი წყება:

```
>>N=64;
>>T=1/128;
>>k=0:N-1;
```

```
>>f=sin(2*pi*20*k*T);
>>plot(k*T,f)
>> grid
>> xlabel('Time, seconds')
>> ylabel('f(kT)')
>> title('20 Hz Sinusoid')
```

f წარმოადგენს 20 ჰერციან სინუსოიდას რომელიც ათვლილია ყოველ 1/128 წამში, რასაც შეესაბამება ათვლის სიხშირე 128 ჰერცი.(სინუსოიდის სიხშირე 20ჰც-ია, ამიტომ ის უნდა ავითვალოთ უფრო დიდი სიხშირით, ვიდრე 40ჰც. 128 ჰერცი ამ შემთხვევაში საკმარისია.)

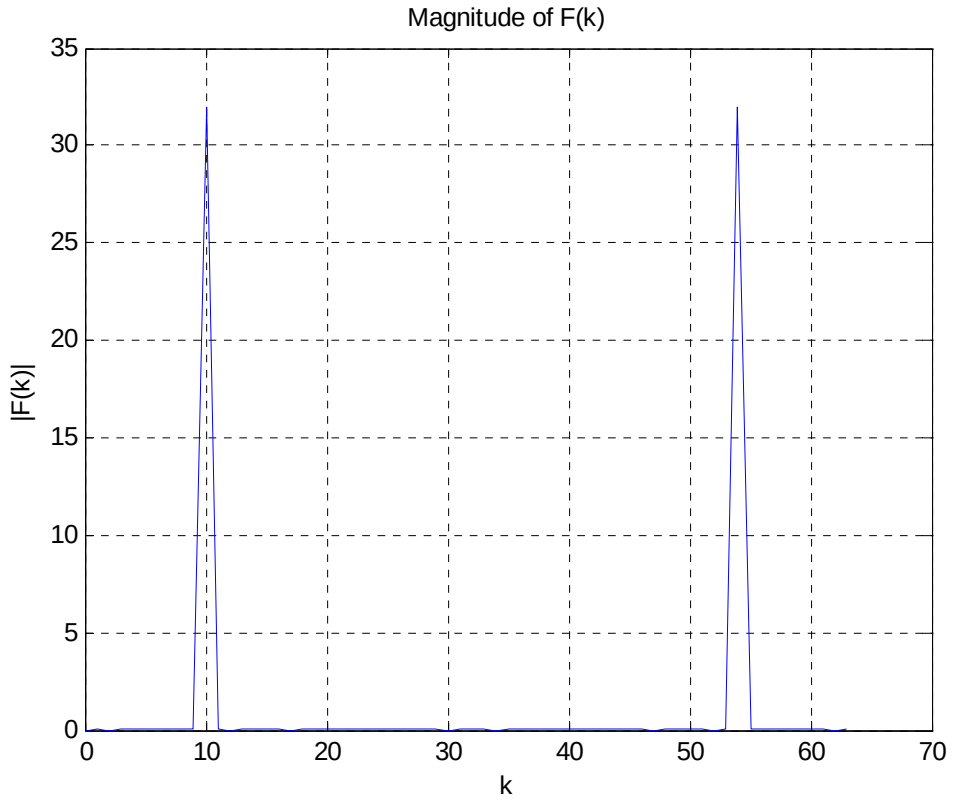


ნახ. 14.2 20 ჰერციანი სინუსოიდა

რადგან ვიცით, რომ სინუსოიდა 20 ჰერციანია, მოსალოდნელია, რომ სიხშირის მნიშვნელობა = 0 ყველგან, გარდა იმ წერტილისა, რომელიც შეესაბამება 20 ჰც. რომ განვსაზღვროთ F_k, რომელიც შეესაბამება 20 ჰც, უნდა გამოვთვალოთ სხვაობა სიხშირულ წერტილებს შორის ჰერცებში, ეს არის 1/NT = 2ჰც.

ამრიგად, 20 ჰერციანი კომპონენტი უნდა გამოჩნდეს F_{10} მნიშვნელობაზე. ნახ. 14.3.

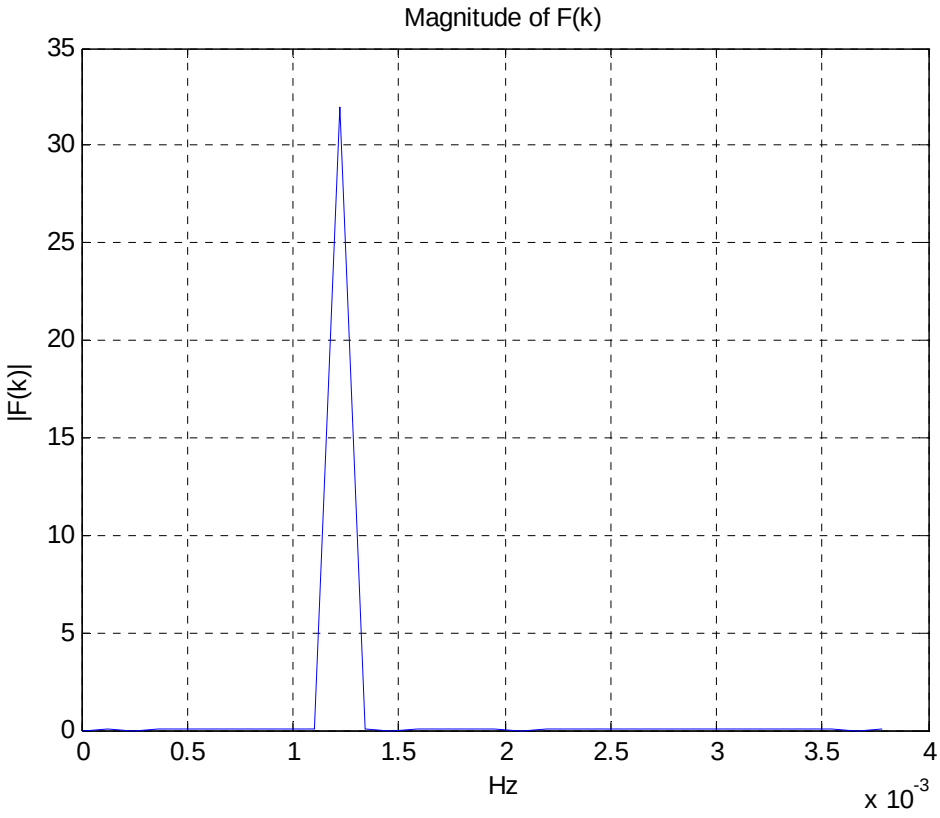
```
>> F=fft(f);
>> magF=abs(F);
>> plot(k, magF),title('Magnitude of F(k)'),...
xlabel('k'), ylabel('|F(k)|'),grid
```



ნახ. 14.3 F_k მოდული

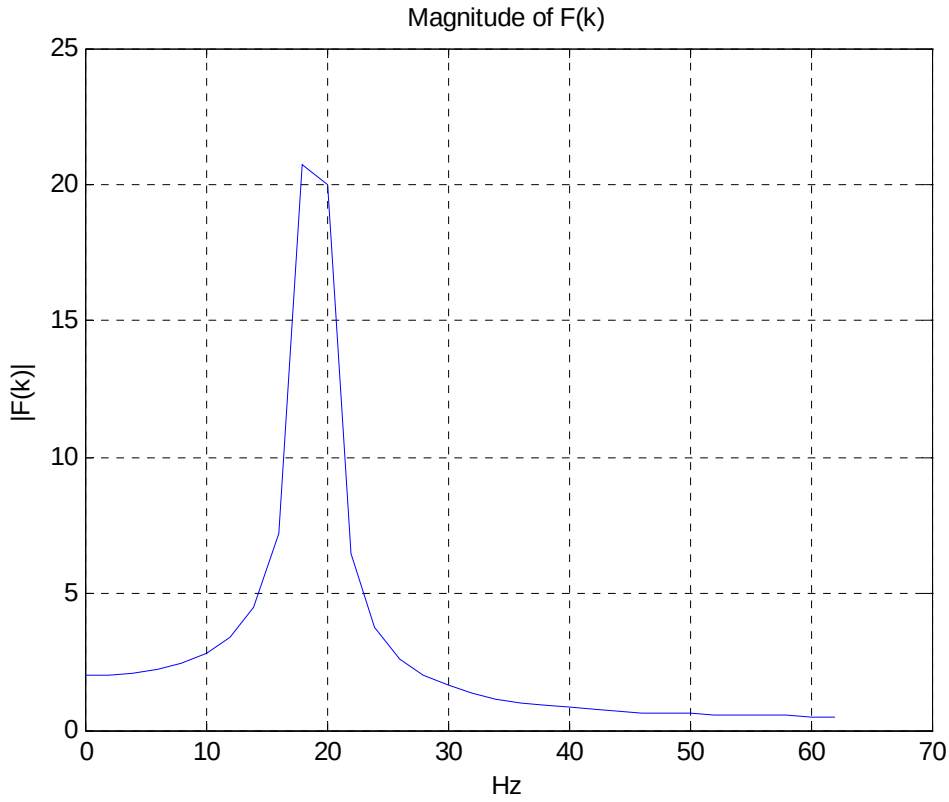
ამ ნახაზზე ჩანს სიმეტრიული ანასახი 20 ჰც კომპონენტისა, რაც გამოწვეულია DFT პერიოდულობით. 20 ჰც კომპონენტი თავს იჩენს როგორც F(54). უნდა აღვნიშნოთ რომ ვექტორი k შეიცავს ინდექსებს, რომელიც შეესაბამება f, F ვექტორთა ელემენტებს. საზოგადოდ რეკომენდებულია აიგოს მონაცემების მხოლოდ პირველი ნახევარი, რომ თავიდან ავიცილოთ სიმეტრიული კომპონენტი. უფრო მოხერხებულია ავაგოთ F_k ჰერცებში გამოსახული სიდიდეებით ნაცვლად k ინდექსებისა.

```
>> hertz = k*(1/N*T);
>> plot(hertz(1:N/2), magF(1:N/2)),...
title('Magnitude of F(k)'),...
>> xlabel('Hz'),ylabel('|F(k)|'),grid
```



ნახ. 14.4 F_k ჰერცებში

დავუშვათ ამ მაგალითში განხილული სინუსოიდის სიხშირე 19 ჰერცია. რადგან სხვაობა ვექტორის ორ მომდევნო მნიშვნელობას შორის 2 ჰერცია, ეს სიხშირე თავს იჩენს k=9.5-ზე, მაგრამ k მნიშვნელობები მთელი რიცხვებია, ამიტომ არ გვაქვს $F_{9.5}$, ამ შემთხვევაში სინუსოიდის სიხშირული კომპონენტი გამოჩნდება F_9 და F_{10} შორის.



ნახ. 14.5 19 ჰერციანი სინუსოიდის სიხშირული მრუდი

ნახ. 14.4 და ნახ. 14.5 ორივე წარმოადგენს ერთი სინუსოიდის სიხშირულ სპექტრს, მაგრამ ერთი მათგანი ხვდება ზუსტად FFT ალგორითმით მიღებულ წერტილზე, მეორე კი არა. ეს არის მაგალითი ე.წ. გაჟონვისა (leakage).

ifft წარმოადგენს ფურიეს შებრუნებული გარდაქმნის ალგორითმს. მისი საშუალებით გამოითვლება დროითი სიგნალი შესაბამისი კომპლექსური სიხშირული მნიშვნელობების მიხედვით. მოცემული ჩანაწერი გამოითვლის F კომპლექსურ მნიშვნელობებს და შემდეგ **ifft** საშუალებით აღადგენს საწყის დროით სიგნალს. საბოლოოდ გამოითვლება სხვაობა სიგნალის საწყის და **ifft-**ს საშუალებით აღდგენილ მნიშვნელობათა შორის, ამ შემთხვევაში იგი ნულის ტოლია.

```
>> N=64;
>> T=1/128;
>> k=0:N-1;
>> f=sin(2*pi*19*k*T);
>> sum(f-ifft(fft(f)))

ans =

 -3.3307e-016
```

FFT ალგორითმი მძლავრი საშუალებაა სიგნალის დამუშავებასა და ანალიზისათვის. ჩვენ განვიხილეთ F_k მოდული, მაგრამ არანაკლებ მნიშვნელოვან ინფორმაციას იძლევა F_k ფაზის გამოთვლა **phase(F_k).**

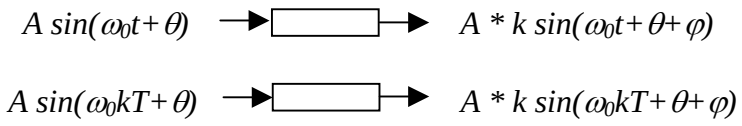
სავარჯიშო

გამოთვალეთ და ააგეთ შემდეგი სიგნალის 128 წერტილის მნიშვნელობა. ააგეთ დროითი სიგნალი. შემდეგ FFT ალგორითმის საშუალებით იპოვეთ სიგნალის სიხშირული მნიშვნელობები და ააგეთ შესაბამისი მრუდი მხოლოდ პირველი 64 წერტილის მიხედვით. ათვლის სიხშირედ აიღეთ 1 კილოჰერცი. შეამოწმეთ მართლაც იმ სიხშირეებზე მიიღეთ თუ არა პიკი, სადაც მოსალოდნელი იყო.

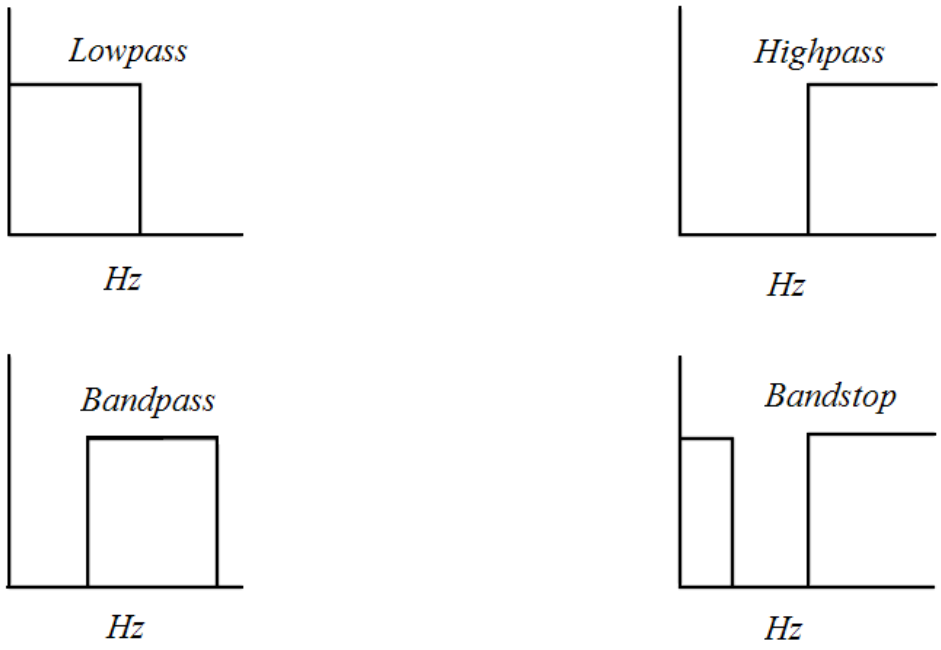
1. $f_k = 2\sin(2\pi 50kT)$
2. $f_k = \cos(250\pi kT) - \sin(200\pi kT)$
3. $f_k = 5 - \cos(1000kT)$
4. $f_k = 4\sin(250\pi kT - \pi/4)$

14.2 ფილტრის ანალიზი

ანალოგური სისტემის გადაცემის(transfer function) ფუნქცია წარმოადგენს კომპლექსურ ფუნქციას *H(s),* ხოლო ციფრული სისტემის გადაცემის ფუნქციაა – *H(z),* ეს ფუნქცია ასახავს სისტემის გავლენას შემოსულ სიგნალზე, ანუ სისტემის ფილტრულ ეფექტს. ორივე ფუნქცია სიხშირის უწყვეტი ფუნქციაა, სადაც $s = j\omega$, $z=e^{j\omega T}$ (გავიხსენოთ, რომ ω არის სიხშირე, გამოსახული რადიანი/წამში.) ამრიგად მოცემულია ω_0, ვუშვებთ, რომ გადაცემის ფუნქციის ამპლიტუდა არის K და ფაზა φ. თუ შემავალი სიგნალი შეიცავს სინუსოიდას ω_0 სიხშირით, გამომავალი სიგნალისათვის ამ სინუსოიდის ამპლიტუდა გამრავლდება K-ზე ფაზა კი გაიზრდება φ-ით.



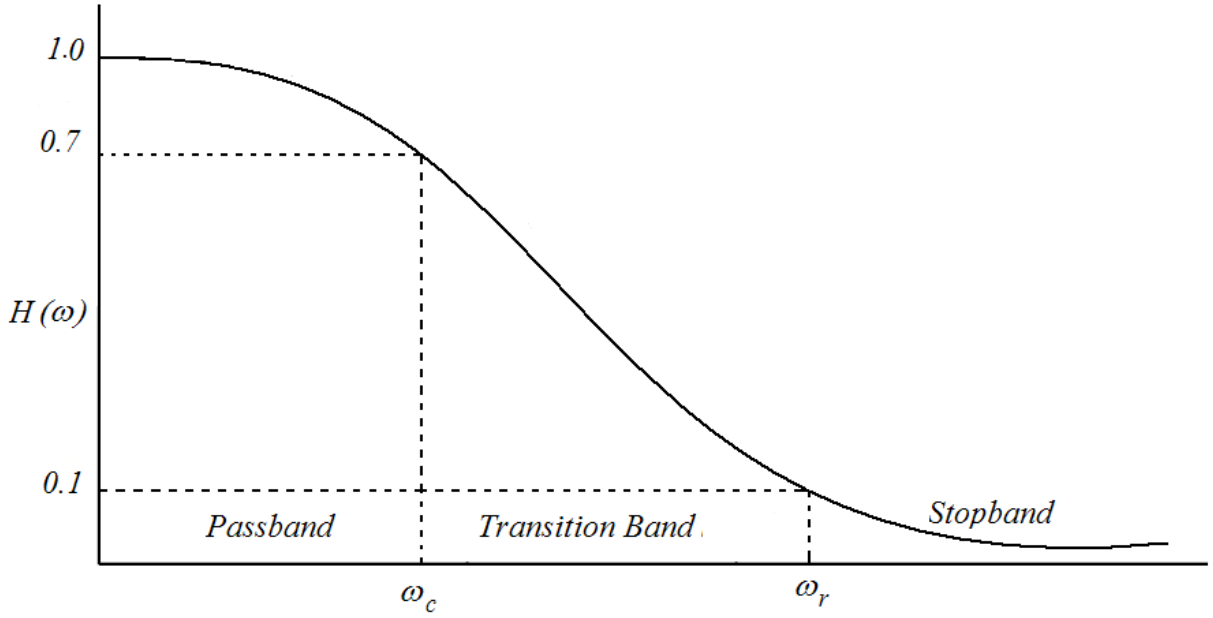
ნახ. 14.6 ფილტრის გავლენა სინუსოიდაზე



ნახ. 14.7 გადაცემის ფუნქცია (იდეალური შემთხვევა)

რადგანაც ფილტრის გადაცემის ფუნქცია განსაზღვრავს ფილტრის ეფექტს სიხშირეზე, იგი უნდა ხასიათდებოდეს სიხშირეთა ზოლით, რომელსაც ის ატარებს. მაგალითად დაბალსიხშირული ფილტრი გაატარებს სიხშირეებს ზღვრულ სიხშირის ქვევით და მოკვეთს დანარჩენს. ზოლოვანი ფილტრი გაატარებს სპექტრის მხოლოდ მოცემულ მონაკვეთს დანარჩენს კი მოკვეთს. bandstop ფილტრი არ გაატარებს სიხშირული სპექტრის მხოლოდ განსაზღვრულ მონაკვეთს. ნახ. ნახ. 14.7-ზე ნაჩვენებია ოთხი ძირითადი ფილტრის გარდაქმნის ფუნქცია იდეალურ შემთხვევაში. პრაქტიკაში შეუძლებელია ასეთი მახასიათებლების მქონე ფილტრის განხორციელება.

მაგალითია ტიპიური დაბალსიხშირული ფილტრისა. გვაქვს 3 უბანი – გატარების, გარდამავალი და მოკვეთის. ეს უბნები განისაზღვრება მოკვეთის ω_c და ω_r გარატების სიხშირეებით. მიღებულია,რომ სიხშირე, რომელიც შეესაბამება ამპლიტუდას 0.7, ჩაითვალოს ზღვრულ(cutoff) სიხშირედ, ხოლო 0.1 ჩამკეტ(rejection) სიხშირედ. ასეთი აღნიშვნის შემოღების შემდეგ, შეგვიძლია ვთქვათ, რომ გატარების ზოლი შეიცავს სიხშირეებს, რომელთა მოდული მეტია ზღვრულ სიხშირეზე, გარდამავალი ზოლი შეიცავს სიხშირეებს, რომელთა მოდული ნაკლებია ზღვრულ (მოკვეთის) და მეტია ჩამკეტ სიხშირეზე, stopband შეიცავს ჩამკეტის მოდულზე ნაკლები მოდულის შესაბამის სიხშირეებს.



ნახ. 14.8 ტიპიური დაბალსიხშირული ფილტრი

რადგან გადაცემის ფუნქცია კომპლექსურია, შესაბამისი ფილტრის ანალიზი ხშირად საჭიროებს მოდულის და ფაზის გრაფიკულ წარმოდგენას. MATLAB-ის ფუნქცია **abs, angle** და **unwrap** შეგვიძლია გამოვიყენოთ კომპლექსური სიდიდეების *H(s), H(z)* მოდულისა და ფაზის გამოსათვლელად. ამასთანავე, ფუნქციები **freqs** და **freqz** გამოიყენება თავად *H(s), H(z)* მნიშვნელობათა გამოსათვლელად.

ანლოგური სისტემის გადაცემის ფუნქცია

ანალოგური ფილტრი განისაზღვრება ფუნქციით *H(s)* სადაც $s = j\omega$. გადაცემის ფუნქციის ზოგადი სახეა:

$$H(s) = \frac{B(s)}{A(s)} = \frac{b_0 s^n + b_1 s^{n-1} + b_2 s^{n-2} + \cdots + b_n}{a_0 s^n + a_1 s^{n-1} + a_2 s^{n-2} + \cdots + a_n}$$

ეს ფუნქცია შეესაბამება *n* რიგის ანალოგურ ფილტრს. მისი ზოგიერთი მაგალითია:

$$H(s) = \frac{0.5279}{s^2 + 1.0275s + 0.5279}$$

$$H(s) = \frac{bs^2}{s^2 + 0.1117s + 0.0062}$$

$$H(s) = \frac{1.05s}{s^2 + 1.05s + 0.447}$$

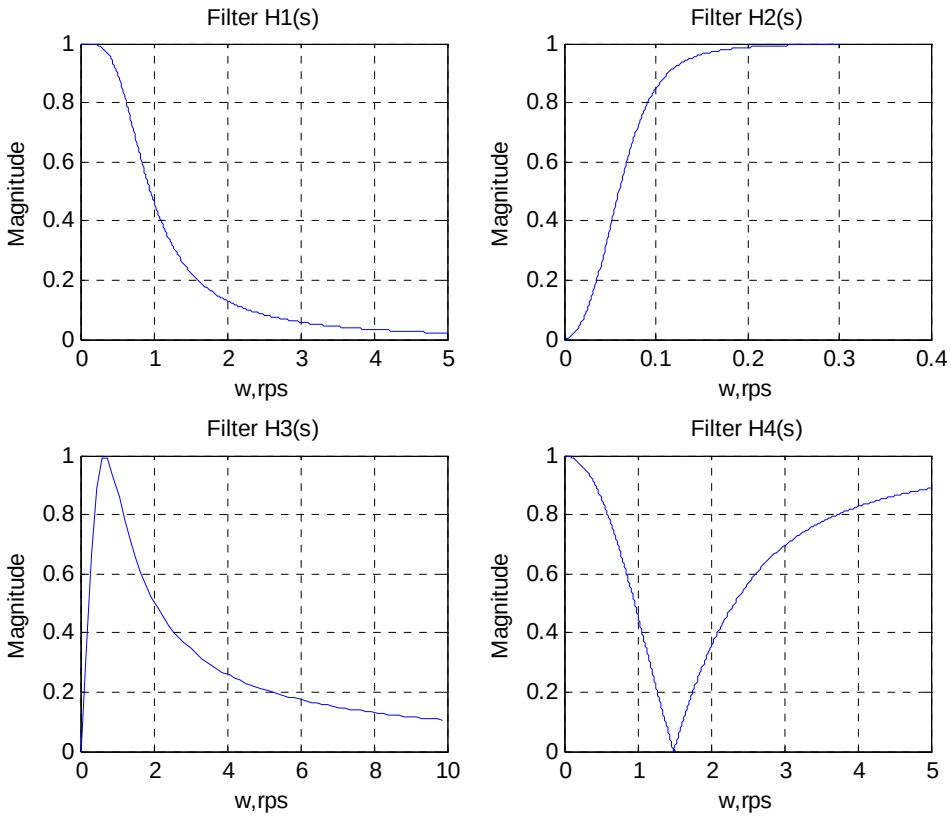
$$H(s) = \frac{s^2 + 2.2359}{s^2 + 2.3511s + 2.2359}$$

იმისათვის, რომ განვსაზღვროთ გადაცემის აქ წარმოდგენილი ფუნქციის მქონე სისტემის მახასიათებლები, ავაგოთ მათი მოდულისა და ფაზის მრუდები. **MATLAB** –ის ფუნქცია

freqs გამოითვლის კომპლექსური ფუნქციის მნიშვნელობებს სამი არგუმენტის საშუალებით. პირველი არგუმენტია ვექტორი, რომელიც შეიცავს *B(s)* პოლინომის კოეფიციენტებს, მეორე - ვექტორი, რომელიც შეიცავს *A(s)* პოლინომის კოეფიციენტებს, ხოლო მესამე - ვექტორი სიხშირეების მნიშვნელობებით რადიანი/წამში. კოეფიციენტების ვექტორები პირდაპირ გადაცემის ფუნქციიდან უნდა ავიღოთ, გარკვეული გამოცდილებაა საჭირო სიხშირეთა ინტერვალის სათანადოდ შესარჩევად. საზოგადოდ, გვჭირდება სიხშირეთა არე, რომელიც იწყება 0 დან და მოიცავს ყველა კრიტიკულ ინფორმაციას ფილტრში. უნდა შეგვეძლოს განვსაზღვროთ ფილტრის ტიპი (დაბალსიხშირული, მაღალსიხშირული, ზოლოვანი, ჩამკეტი stopband) და კრიტიკული სიხშირეები ზღვრული და ჩამკეტი).

ქვემოთ მოყვანილი პროგრამა განსაზღვრავს და აგებს გადაცემის 4 ფუნქციის მახასიათებელ მრუდებს

```
%     This programm datermines and plots the
%     magnitudes of four analog filters
%
w1 = 0:0.05:5;
B1 = [0.5279];
A1 = [1,1.0275,0.5279];
H1s = freqs(B1,A1,w1);
%
w2 = 0:0.001:0.3;
B2 = [1,0,0];
A2 = [1,0.1117,0.0062];
H2s = freqs(B2,A2,w2);
%
w3 = 0:0.15:10;
B3 = [1.05,0];
A3 = [1,1.05,0.447];
H3s = freqs(B3,A3,w3);
%
w4 = 0:0.005:5;
B4 = [1,0,2.2359];
A4 = [1,2.3511,2.2359];
H4s = freqs(B4,A4,w4);
clf
subplot(221),plot(w1,abs(H1s)),title('Filter H1(s)'),...
    xlabel('w,rps'),ylabel('Magnitude'),grid
subplot(222),plot(w2,abs(H2s)),title('Filter H2(s)'),...
    xlabel('w,rps'),ylabel('Magnitude'),grid
subplot(223),plot(w3,abs(H3s)),title('Filter H3(s)'),...
    xlabel('w,rps'),ylabel('Magnitude'),grid
subplot(224),plot(w4,abs(H4s)),title('Filter H4(s)'),...
    xlabel('w,rps'),ylabel('Magnitude'),grid
```



ნახ. 14.9 სხვადასხვა ტიპის ანალოგური ფილტრების მახასიათებელი მრუდები

ფილტრის ფაზები შეგვიძლია ავაგოთ **angle** ან **unwrap** ფუნქციის გამოყენებით. **angle** გვაძლევს კუთხეს რადიანებში მხოლოდ [-π,π] ინტერვალში. ამ ხარვეზის თავიდან ასაცილებლად ვსარგებლობთ ფუნქციით **unwrap** ასეთი სახით unwrap(angle(w)).

14.3 დისკრეტული გადამცემი ფუნქცია

დისკრეტული ფილტრი განისაზღვრება გადაცემის კომპლექსური ფუნქციით *H(z)*, სადაც $z=e^{j\omega T}$. z შეიძლება ჩაიწეროს როგორც სიხშირის (ω) ან ნორმირებული სიხშირის (ωT) ფუნქცია, თუ z სიხშირის ფუნქციაა, *H(z)* აგრეთვე სიხშირის ფუნქცია იქნება. თუკი *H(z)* მოვარგეთ (applied) სიგნალს ათვლის ინტერვალით T, სიხშირეების შესაბამისი არე იქნება 0 დან ნაიქვისტის სიხშირემდე, რომელიც ტოლია π/T რადიანი/წმ, ან 1/2T ჰერცი. თუ დავუშვებთ, რომ z არის ნორმირებული სიხშირის ფუნქცია, *H(z)* ექნება სიხშირეთა შესაბამისი ინტერვალი 0 დან π-მდე.

H(z) გადაცემის ფუნქციის ზოგადი სახეა:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}$$

ეს ფუნქცია შეესაბამება *n* რიგის ციფრულ ფილტრს. განვიხილოთ ზოგიერთი მაგალითი:

$$H(z) = \frac{0.2066 + 0.4131z^{-1} + 0.2066z^{-2}}{1 - 0.3695z^{-1} + 0.1958z^{-2}}$$
$$H(z) = \frac{0.894 - 1.789z^{-1} + 0.894z^{-2}}{1 - 0.778z^{-1} + 0.799z^{-2}}$$
$$H(z) = \frac{0.42 - 0.42z^{-2}}{1 - 0.44z^{-1} + 0.159z^{-2}}$$
$$H(z) = \frac{0.5792 + 0.4425z^{-1} + 0.5792z^{-2}}{1 + 0.4425z^{-1} + 0.1584z^{-2}}$$

თუ ფუნქცია ჩაწერილია როგორც z-ის დადებითი ხარისხების ფუნქცია, მრიცხველი და მნიშვნელი უნდა გავყოთ z სათანადო ხარისხზე, რომ მივიღოთ ზემოთ მოყვანილის ანალოგიური გამოსახულება.

ციფრული ფილტრი შესაძლოა წარმოდგენილი იყოს სტანდარტული სხვაობითი განტოლების სახით (standard difference equation). მას აქვს ზოგადი სახე:

$$y_n = \sum_{k=-N_1}^{N_2} b_k x_{n-k} - \sum_{k=1}^{N_3} a_k y_{n-k}$$

არსებობს პირდაპირი კავშირი გადამცემ განტოლებასა და სტანდარტულ სხვაობით განტოლებას სორის თუ დავუშვებთ, რომ $N_1=0$,

$$y_n = \sum_{k=0}^{N_2} b_k x_{n-k} - \sum_{k=1}^{N_3} a_k y_{n-k}$$

ასეთ ფორმაში a_k და b_k კოეფიციენტები წარმოადგენს გადაცემის ფუნქციის კოეფიციენტებს a_0 = 1-ით. ამრიგად განტოლება, რომელიც შეესაბამება პირველ მაგალითს, ასე ჩაიწერება:

$$y_n = 0.2066x_n + 0.4131x_{n-1} + 0.2066x_{n-2} + 0.3695y_{n-1} - 0.1958y_{n-2}$$

თუ a_k ყველა კოეფიციენტი 0-ის ტოლია, გარდა a_0, რომელიც 1-ის ტოლია, მაშინ შესაბამისი გადაცემის ფუნქციის მნიშვნელი ტოლია 1-ის:

$$y_n = 0.5x_n - 1.2x_{n-1} + 0.25x_{n-3}$$
$$H(z) = 0.5 - 1.2z^{-1} + 0.25z^{-3}$$

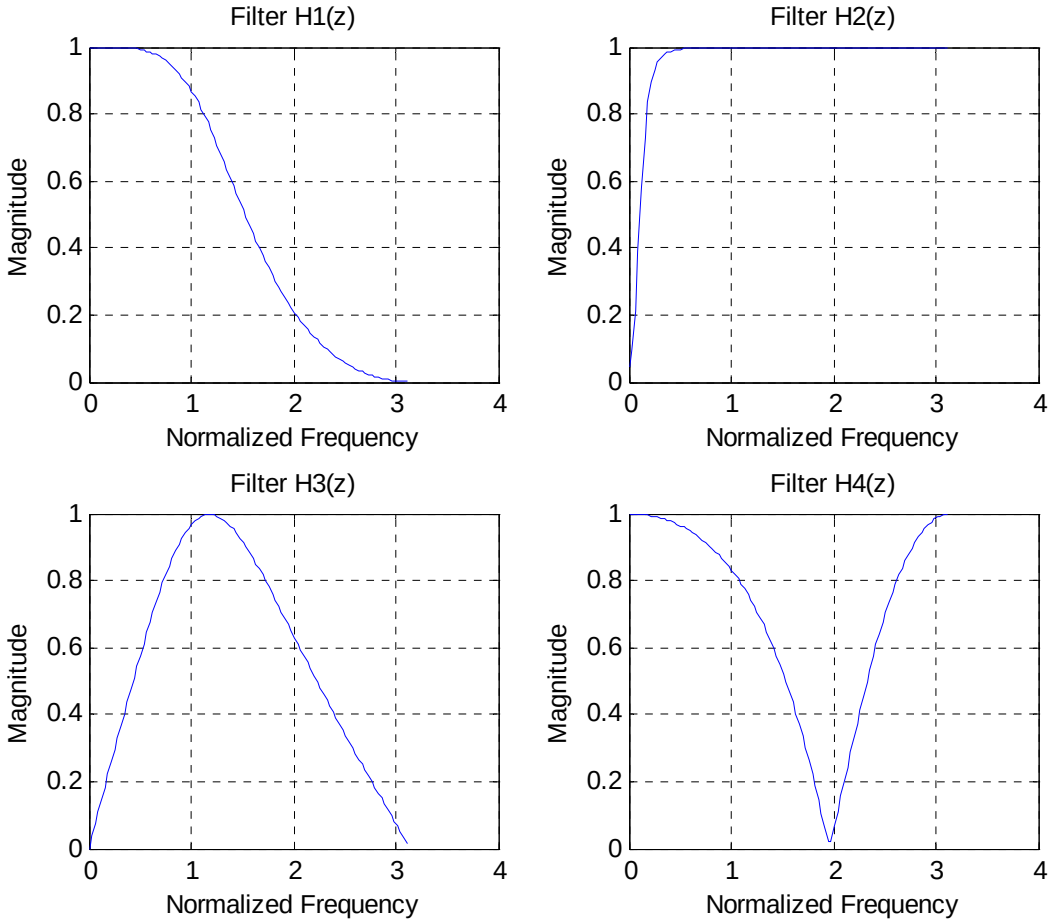
თუ გადაცემის ფუნქციის მნიშვნელი 1-ის ტოლია, მოცემული გვაქვს ფილტრი სასრული იმპულსური მახასიათებლით (FIR-finite impulse response) , თუ არა - უსასრულო იმპულსური მახასიათებლით (IIR- infinite impulse response) . ფილტრის ორივე ტიპი ფართოდ გამოიყენება სიგნალის დამუშავებაში.

იმისათვის რომ გავიგოთ მოცემული გადაცემის ფუნქციის მქონე სისტემის მახასიათებლები, უნდა ავაგოთ მისი მახასიათებელი მრუდი. **MATLAB** –ს ფუნქცია **freqz** გამოითვლის კომპლექსური ფუნქციის მნიშვნელობებს სამი არგუმენტის საშუალებით. პირველი არგუმენტია ვექტორი, რომელიც შეიცავს *B(z)* პოლინომის კოეფიციენტებს, მეორე არგუმენტია ვექტორი, რომელიც შეიცავს *A(z)* პოლინომის კოეფიციენტებს, ხოლო მესამე

არგუმენტია მთელი რიცხვი, რომელიც განსაზღვრავს ნორმირებულ სიხშირეთა რაოდენობას, რომელიც გამოყენებულია $[0.\ \pi]$ ინტერვალზე. კოეფიციენტები განისაზღვრება ფუნქციიდან, ხოლო წერტილების რაოდენობა, რომელიც გარდაქმნის ფუნქციის მნიშვნელობათა გამოსათვლელად გამოიყენება განსაზღვრავს გარჩევას, რომელიც ისე უნდა შეირჩეს რომ შევძლოთ ფილტრის ტიპის დადგენა.

პროგრამა გამოითვლის და აგებს მახასიათებელ მრუდებს ზემოთგანხილული 4 ფილტრისათვის:

```
%     This programm datermines and plots the
%     magnitudes of four digital filters
%

B1 = [0.2066,0.4131,0.2066];
A1 = [1,-0.3695,0.1958];
[H1z,w1T] = freqz(B1,A1,100);
%

B2 = [0.894,-1.789,0.894];
A2 = [1,-1.778,0.799];
[H2z,w2T] = freqz(B2,A2,100);
%

B3 = [0.42,0,-0.42];
A3 = [1,-0.443,0.159];
[H3z,w3T] = freqz(B3,A3,100);
%

B4 = [0.5792,0.4425,0.5792];
A4 = [1,0.4425,0.1584];
[H4z,w4T] = freqz(B4,A4,100);
clf
subplot(221),plot(w1T,abs(H1z)),title('Filter H1(z)'),...
    xlabel('Normalized Frequency'),ylabel('Magnitude'),grid
subplot(222),plot(w2T,abs(H2z)),title('Filter H2(z)'),...
    xlabel('Normalized Frequency'),ylabel('Magnitude'),grid
subplot(223),plot(w3T,abs(H3z)),title('Filter H3(z)'),...
    xlabel('Normalized Frequency'),ylabel('Magnitude'),grid
subplot(224),plot(w4T,abs(H4z)),title('Filter H4(z)'),...
    xlabel('Normalized Frequency'),ylabel('Magnitude'),grid
```



ნახ. 14.10 სხვადასხვა ტიპის ციფრული ფილტრების მახასიათებელი მრუდები

MATLAB-ს აქვს აგრეთვე ფუნქცია **grpdelay**, რომელიც გამოიყენება ციფრული ფილტრის ჯგუფური შეყოვნების განსასაზღვრავად. ჯგუფური შეყოვნება ეს არის ფილტრის, როგორც სიხშირის ფუნქციის საშუალო შეყოვნების ზომა. ის განისაზღვრება როგორც ფილტრის ფაზური მახასიათებლის უარყოფითი პირველი წარმოებული. თუ გადაცემის ფუნქციის ფაზური მახასიათებელია $\theta(\omega)$, მაშინ ჯგუფური შეყოვნება იქნება:

$$\tau(\omega) = -\frac{d\theta(\omega)}{d\omega}$$

grpdelay ფუნქციას აქვს 3 არგუმენტი: B(z) და A(z) კოეფიციენტები და მთელი რიცხვი, რომელიც განსაზღვრავს შეყოვნების ზომას ნორმირებული სიხშირის 0 - π ინტერვალზე. რამდენადაც ეს ფუნქცია იყენებს **fft** ფუნქციას, სასურველია მესამე არგუმენტის მნიშვნელობა შეირჩეს ისე, რომ იყოს 2-ის ხარისხის ტოლი.

რაციონალურ წილადთა ჯამად დაშლა

ანალოგური და ციფრული ფილტრების ანალიზისას ხშირად გვჭირდება გადაცემის ფუნქციის დაშლა რაციონალურ წილადთა ჯამად. ეს პროცესი შეიძლება გამოვიყენოთ იმისათვის, რომ ფილტრი წარმოვადგინოთ ქვეფუნქციათა კასკადური ან პარალელური სტრუქტურის სახით. გარდა ამისა რაციონალურ წილადად დაშლა შეიძლება გამოვიყენოთ იმისათვის, რომ

შევასრულოთ უკუგარდაქმნა სიხშირულიდან დროით გარემოში. **MATLAB**-ს აქვს ფუნქცია **residue** რომელიც ახდენს ორი პოლინომის განაყოფის რაციონალურ წილადთა ჯამად დაშლას. ეს ფუნქცია შეგვიძლია გამოვიყენოთ გადაცემის ფუნქციისათვის, რადგან იგი სწორედ ორი პოლინომის განაყოფის სახითაა წარმოდგენილი.

დავუშვათ G არის υ ცვლადიანი ორი პოლინომის განაყოფი. ის შეიძლება წარმოვადგინოთ როგორც შერეული წილადი;

$$G(\upsilon)=\frac{B(\upsilon)}{A(\upsilon)}=\sum_{n=0}^{N}k_n\upsilon^n+\frac{N(\upsilon)}{D(\upsilon)}$$

წილადური ნაწილი არის ორი პოლინომის განაყოფი. მისი მრიცხველი შესაძლოა წარმოვადგინოთ როგორც რამდენიმე თანამამრავლის ნამრავლი, რომლებიც უდრის ამ პოლინომის ცვლადისა და მისი ფესვების სხვაობას. (მრიცხველი პოლინომის ფესვებს ეწოდება ფუნქციის ნულები, ხოლო მნიშვნელისას – პოლუსები).

$$\frac{N(\upsilon)}{D(\upsilon)}=\frac{b_1\upsilon^{n-1}+b_2\upsilon^{n-2}+\cdots+b_{n-1}\upsilon+b_1}{(\upsilon-p_1)^{m_1}(\upsilon-p_1)^{m_2}\cdots(\upsilon-p_1)^{m_r}}$$

რომელიც შეგვიძლია ჯამის სახით წარმოვადგინოთ;

$$\frac{N(\upsilon)}{D(\upsilon)}=\frac{C_{1,1}}{\upsilon-p_1}+\frac{C_{1,2}}{(\upsilon-p_1)^2}+\cdots+\frac{C_{1,m_1}}{(\upsilon-p_2)^{m_1}}+\frac{C_{2,1}}{\upsilon-p_2}+\frac{C_{2,2}}{(\upsilon-p_2)^2}+\cdots+\frac{C_{2,m_1}}{(\upsilon-p_2)^{m_1}}$$
$$+\frac{C_{r,1}}{\upsilon-p_r}+\frac{C_{r,2}}{(\upsilon-p_r)^2}+\cdots+\frac{C_{r,m_1}}{(\upsilon-p_r)^{m_r}}$$

ფუნქციას **residue** აქვს ორი **input** მნიშვნელობა: A და B პოლინომის კოეფიციენტები., გამოითვლის და გვაძლევს სამ ვექტორს – *r, p, k*. *r* ვექტორი შეიცავს $C_{i,j}$ კოეფიციენტებს, *p* – პოლუსების მნიშვნელობებს p_n და *k* – k_n მნიშვნელობებს. იმისათვის რომ სწორი ინტერპრეტაცია მივცეთ **residue** ფუნქციის შედეგად მიღებულ მნიშვნელობებს, კარგად უნდა დავუკვირდეთ ფორმულაში შემოღებული აღნიშვნების მნიშვნელობას. მნიშვნელოვანია გავითვალისწინოთ, რომ მოცემული წილადი რაციონალური წილადების ჯამის სახით რამდენიმეგვარად შეიძლება წარმოვადგინოთ. თუმცა აქვე აღვნიშნავთ, რომ ეს ფუნქცია ყოველთვის ერთიდაიგივე შედეგს მოგვცემს მრიცხველისა და მნიშვნელის მოცემული წყვილისათვის.

ფუნქციის საილუსტრაციოდ განვიხილოთ მაგალითი:

$$F(z)=\frac{z^2}{z^2-1.5z+0.5}$$

MATLAB საშუალებით იგი შეგვიძლია დავშალოთ რაციონალური წილადების ჯამად:

```
B=[1, 0, 0];
A=[1, -1.5, 0.5];
[r,p,k]=residue(B,A)
```

შედეგად მივიღებთ:

$$r = \begin{bmatrix} 2.0 \\ -0.5 \end{bmatrix} \quad p = \begin{bmatrix} 1.0 \\ 0.5 \end{bmatrix} \quad k = [1]$$

ამიტომ გვექნება:

$$F(z) = \frac{z^2}{z^2 - 1.5z + 0.5} = 1 + \frac{2}{z - 1.0} - \frac{0.5}{z - 0.5}$$

განვიხილოთ შემდეგი წილადი:

$$H(z) = \frac{1}{z^2 - 3.5z + 1.5}$$

```
B=[1];
A=[1, -3.5, 1.5];
[r,p,k]=residue(B,A)
```

შედეგად მივიღებთ:

$$r = \begin{bmatrix} 0.4 \\ -0.4 \end{bmatrix} \quad p = \begin{bmatrix} 3.0 \\ 0.5 \end{bmatrix} \quad k = [\ ]$$

ესე იგი გვექნება:

$$F(z) = \frac{1}{z^2 - 3.5z + 1.5} = \frac{0.4}{z - 3} - \frac{0.4}{z - 0.5}$$

თუ გვსურს ჩავწეროთ ფორმულა z უარყოფითი ხარისხებით, წილადის მრიცხველი და მნიშვნელი გავამრავლოთ z^{-1}:

$$F(z) = \frac{0.4z^{-1}}{1 - 3z^{-1}} - \frac{0.4z^{-1}}{1 - 0.5z^{-1}}$$

სავარჯიშო

თითოეული ფუნქციისათვის ააგეთ მახასიათებელი მრუდი (მაგნიტუდე). განსაზღვრეთ ფილტრის ტიპი და მახასიათებლები. ციფრული ფილტრებისათვის ისარგებლეთ ნორმირებული სიხშირით.

1. $H(s) = \dfrac{s^2}{s^2 + \sqrt{2}s + 1}$
2. $H(z) = \dfrac{0.707z - 0.707}{z - 0.414}$
3. $H(z) = -0.163 - 0.058z^{-1} + 0.116z^{-2} + 0.2z^{-3} + 0.116z^{-4} - 0.058z^{-5} - 0.163z^{-6}$
4. $H(s) = \dfrac{5s + 1}{s^2 + 0.4s + 1}$

14.4 ციფრული ფილტრის განხორციელება

ანალოგური ფილტრები ხორციელდება მოწყობილობებში ისეთი კომპონენტების გამოყენებით, როგორიცაა რეზისტორი და კონდენსატორი. ციფრული ფილტრი ხორციელდება პროგრამულად. ციფრული ფილტრი შეგვიძლია განვახორციელოთ როგორც გადაცემის ფუნქცია $\boldsymbol{H_z}$ ან სტანდარტული difference განტოლება. ფუნქციის **filter** საწყისი არგუმენტია სიგნალი და საბოლოო შედეგიც სიგნალია, რომელმაც გაიარა ფილტრი. განტოლება განსაზღვრავს შემავალის გამომავალ სიგნალად გარდაქმნის საფეხურებს.

დამოკიდებულება შემავალსა x_n და გამომავალ y_n სიგნალს შორის ასეთია:

$$y_n = \sum_{k=-N_1}^{N_2} b_k x_{n-k} - \sum_{k=1}^{N_3} a_k y_{n-k}$$

მაგალითად:

$$y_n = 0.04x_{n-1} + 0.17x_{n-2} + 0.25x_{n-3} + 0.17x_{n-4} + 0.04x_{n-5}$$
$$y_n = 0.42x_n - 0.42x_{n-2} + 0.44x_{n-1} - 0.16x_{n-2}$$
$$y_n = 0.33x_{n+1} + 0.33x_n + 0.33x_{n-1}$$

პირველი ფილტრის გამომავალი სიგნალი დამოკიდებულია შემავალი სიგნალის მხოლოდ წარსულ მნიშვნელობებზე. მაგალითად რომ გამოვთვალოთ y_{10} უნდა იცოდეთ x_9, x_8, x_7, x_6, x_5 მნიშვნელობები. ამ ტიპის ფილტრი წარმოადგენს *FIR* ფილტრს და მისი გადამცემი ფუნქციის მნიშვნელი 1-ის ტოლია. მეორე ფილტრი მოითხოვს არა მარტო შემავალი სიგნალის მნიშვნელობებს, არამედ გამომავალი სიგნალის წრსულ მნისვნელობებსაც, რომ გამოითვალოს ახალი გამომავალი სიგნალი. ამ ტიპის ფილტრს უწოდებენ *IIR* ფილტრს. მესამე ფილტრი არის აგრეთვე *FIR* ფილტრი, მაგრამ გამომავალი სიგნალი დამოკიდებულია მხოლოდ შემავალ სიგნალზე. თუმცა უნდა შევნიშნოთ, რომ ინდექსები მესამე განტოლებაში მოითხოვს, რომ შეგვეძლოს წინასწარ განვჭვრიტოთ შემავალი სიგნალის მნიშვნელობები. ასე, რომ თუ გვინდა გამოვითვალოთ y_5 უნდა ვიცოდეთ x_6, x_5, x_4.
ეს მოთხოვნა პრობლემას არ წარმოადგენს, თუ შემავალ სიგნალს ვითვლით ფუნქციის საშუალებით, ან მისი მნიშვნელობები მოცემულია ფაილის სახით. მაგრამ პრბლემა შეიქმნება, თუ ეს სიგნალი იქმნება რეალურად, ექსპერიმენტის მიმდინარეობისას.

MATLAB-ში სიგნალის 'გაფილტრვა' ხდება ფუნქციით **filter**. იგულისხმება, რომ სტანდარტულ difference განტოლებას აქვს სახე:

$$y_n = \sum_{k=0}^{N_2} b_k x_{n-k} - \sum_{k=1}^{N_3} a_k y_{n-k}$$

რაც შეესაბამება გადაცემის ფუნქციას:

$$H(z) = \frac{B(s)}{A(s)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}$$

filter ფუნქციის პირველი 2 არგუმენტია $[a_k]$ და $[b_k]$ კოეფიციენტების ვექტორი.

მესამე არგუმენტია შემავალი სიგნალი.

```
>>B=[0.0, 0.04, 0.17, 0.25, 0.17, 0.04];
>>A=[1];
>>y=filter(B,A,x);
```

ეს ფუნქცია არ შეგვიძლია გამოვიყენოთ მესამე ფილტრისათვის, რადგან დიფერენციალურ განტონებას შეუძლია გამოითვალოს სიგნალის მნიშვნელობები მხოლოდ k=0 დან. მესამე განტოლებაში მოითხოვება, რომ პირველი ჯამი დაიწყოს k=-1 მნიშვნელობით. ამ შემთხვევაში ფილტრს 'მოვარგებთ' ე.წ. ვექტორული არითმეტიკის საშუალებით. დავუშვათ,რომ შემავალი სიგნალი წარმოდგენილია x ვექტორის სახით. შესაბამისი გამომავალი y სიგნალი შეგვიძლია გამოვითვალოთ შემდეგი ბრძანებების საშუალებით:

```
N=length(x);
y(1)=0.33*x(1)+0.33*x(2);
for n=2:N-1
      y(n) = 0.33*x(n-1) + 0.33*x(n) + 0.33*x(n-1);
end
y(N) = 0.33*x(N-1) + 0.33*x(N);
```

ჩვენ დავუშვით, რომ x-ის ის მნიშვნელობები, რომელთათვისაც არ გვაქვს სიდიდეები მოცემული, 0-ის ტოლია [x(-1) და x(N+1)] . გამოთვლის სხვა გზაც არსებობს:

```
N=length(x);
y(1)=0.33*x(1)+0.33*x(2);
y(2:N-1) = 0.33*x(3:N) + 0.33*x(2:N-1) + 0.33*x(1:N-2);
y(N)=0.33*x(N-1) + 0.33*x(N);
```

ნებისმიერი ფილტრი შეგვიძლია განვახორციელოთ ვექტორული ოპერაციებით, მაგრამ ფუნქცია ყოველთვის იძლევა უფრო მარტივ ამოხსნას.

ფილტერ ფუნქცია იძლევა ორ გამოსავალ არგუმენტს:

```
[y, state] = filter(b,a,x)
```

y ვექტორი შეიცავს გამომავალ არგუმენტს, როცა *x* ვექტორი – საწყისი სიგნალია, ხოლო ვექტორი state სიდიდეთა საბოლოო მწკრივს, რომელსაც ფილტრი იყენებს. მაგალითად, თუ გვინდა გავფილტროთ x2 ვექტორი, რომელიც x სიგნალის სხვა სეგმენტს წარმოადგენს, შეგვიძლია მივუთითოთ, რომ საწყისი პირობები განსაზღვრულია ვექტორში state, ამ შემთხვევაში x და x2 ვექტორები განიხილება როგორც ერთი გრძელი ვექტორი ნაცვლად 2 სხვადასხვა ვექტორისა.

```
y  = filter(b,a,x2,state);
```

ბოლოს, ფუნქცია **conv** შეგვიძლია გამოვიყენოთ, იმისათვის, რომ გამოვითვალოთ FIR ფილტრის გამოსავალი მნიშვნელობები. ეს ფუნქცია გამოიყენება მხოლოდ FIR ფილტრისათვის. უმჯობესია **conv** გამოვიყენოთ პოლინომების გადამრავლებისათვის, ხოლო **filter** სიგნალის გასაფილტრად.

სავარჯიშო

შემდეგი გადაცემის ფუნქცია შეიქმნა იმისათვის, რომ ფილტრმა გაატაროს სიგნალი სიხშირეებზე 500 – 1500 ჰერცს შორის. სიგნალი ათვლილია 5 კილოჰერცი სიხშირით:

$$H(z) = \frac{0.42z^2 - 0.42}{z^2 - 0.443z + 1.59}$$

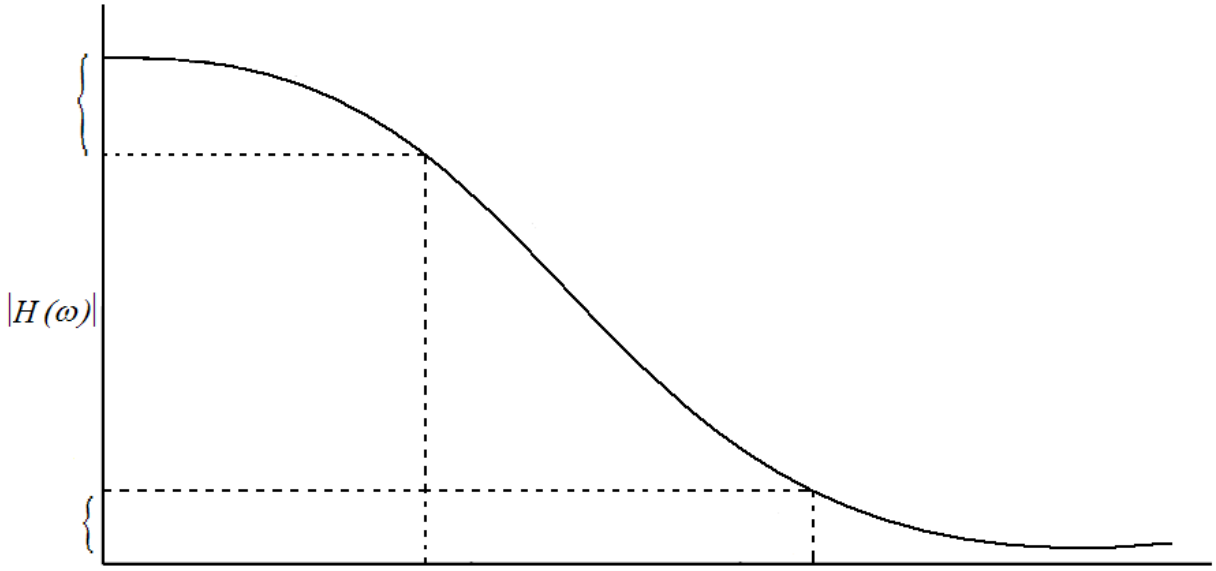
აიღეთ შემდეგი სიგნალი, გამოთვალეთ როგორი იქნება იგი ფილტრში გავლის შემდეგ. ააგეთ ორივე სიგნალის სიხშირული მრუდი ერთიოდაიგივე ნახაზზე, რომ თვალსაჩინოდ გამოჩნდეს ფილტრის ეფექტი.

1. $x_k = \sin(2\pi 1000kT)$
2. $x_k = 2\cos(2\pi 100kT)$
3. $x_k = -\sin(2\pi 200kT)$
4. $x_k = \cos(2\pi 1600kT)$

14.5 ციფრული ფილტრის დიზაინი

IIR (infinit impulse response filter) (ფილტრი უსასრულო იმპულსური მახალიათებლით)

MATLAB –ს აქვს ფუნქციები ოთხი სხვადასხვა ტიპის ციფრული ფილტრისათვის, რომელიც ეყრდნობა ანლოგური ფილტრის დიზაინს. Butterworth ფილტრს აქვს ბრტყელი მახასიათებელი მრუდი ბრტყელი გატარების(ზღვრული) და მოკვეთის(ცუტოფფ) არეებით. Chebishev-ის და ელიფსური ფილტრების მახასიათებელი მრუდი კი ტალღოვანია თითქმის მთელ უბანზე. მაგრამ Chebishev-ის ფილტრს აქვს ყველაზე ვიწრო გარდამავალი უბანი, რაც მის უპირატესობას წარმოადგენს სხვა ფილტრებთან შედარებით.



ნახ. 14.11 გადაცემის ფუნქციის ტალღოვანი უბნები

MMATLAB-ის ფუნქციებს დაბალსიხშირული IIR ფილტრებისათვის ანალოგურ პროტოტიპზე დაყრდნობით შემდეგი სახე აქვს:

```
[B,A] = butter(N,Wn);
[B,A] = cheby1(N,Rp,Wn);
[B,A] = cheby2(N,Rs,Wn);
[B,A] = ellip(N,Rp,Rs,Wn);
```

ფუნქციის შესავალ არგუმენტებს წარმოადგენს ფილტრის რიგი (N), (Rp , Rs) ripple და ნორმირებული მომკვეთი (cutoff) სიხშირე(Wn). ნორმირებული სიხშირე ემყარება ნაიქვისტის სიხშირეს, რომელიც 1-ის ტოლად მიიჩნევა. (აღვნიშნავთ, რომ ეს განსხვავდება **freqz** ფუნქციის ნორმირებული სიხშირისაგან). შედეგად მიღებული ვექტორები B და A არის გადაცემის ფუნქციის კოეფიციენტები.

რომ შევქმნათ ზოლოვანი ფილტრი, არგუმენტები ისეთივე უნდა გამოვიყენოთ, როგორც დაბალსიხშირული ფილტრისას, მხოლოდ Wn იქნება ორელემენტიანი ვექტორი, რომელიც ახასიათებს ზოლის საწყის და საბოლოო ნორმირებულ სიხშირეებს (Wn(1)-Wn(2)).

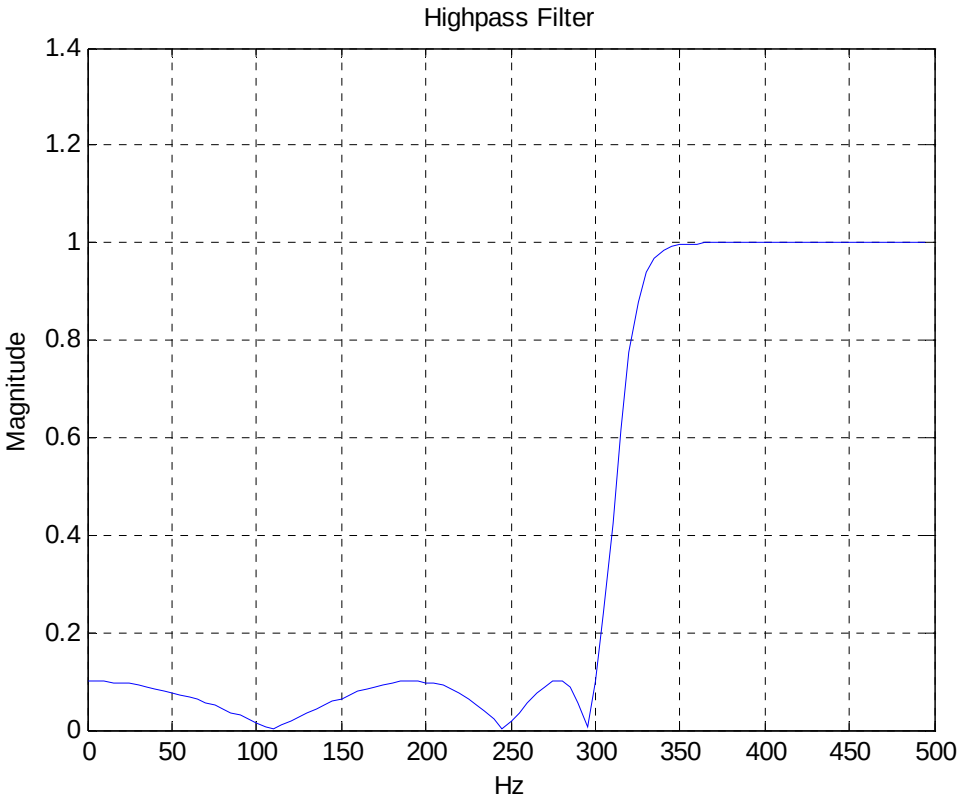
მაღალსიხშირული ფილტრის შესაქმნელად გვჭირდება დამატებითი პარამეტრი ‘high’,

```
[B,A] = butter(N,Wn,’high’);
[B,A] = cheby1(N,Rp,Wn, ’high’);
[B,A] = cheby2(N,Rs,Wn, ’high’);
[B,A] = ellip(N,Rp,Rs,Wn, ’high’);
```

bandstop ფილტრის შესაქმნელად არგუმენტები იგივე რჩება, მხოლოდ ნაცვლად ‘high’ იწერება ‘stop’, თანაც Wn არგუმენტი უნდა იყოს ორელემენტიანი ვექტორი, რომელიც ზოლის საზღვრებს ახასიათებს.

საილუსტრაციოდ, დავუშვათ გვინდა შევქმნათ მე-6 რიგის ჩებიშევის II ტიპის ფილტრი. ვთქვათ ასევე გვინდა შევზღუდოთ გატარების ტალღა(ripple) 0.1-ით (20 დეციბელი). ფილტრი გვინდა გამოვიყენოთ სიგნალისათვის, რომელიც აითვლება 1კჰერცზე, რაც ნიშნავს, რომ ნაიქვისტის სიხშირე 500 ჰერცია. მოკვეთის cutoff სიხშირე უნდა იყოს 300 ჰერცი, ე.ი. ნორმირებული სიხშირეა 300/500, ანუ 0.6.

```
[B,A] = cheby2(6,20,0.6, 'high');
[H,wT] = freqz(B,A,100);
T = 0.001;
hertz = wT/(2*pi*T);
plot(hertz, abs(H)), title('Highpass Filter'),...
xlabel('Hz'),ylabel('Magnitude'), grid
```



ნახ. 14.12 ჩებიშევის II ტიპის ფილტრი

ეს ფილტრი რომ სიგნალს მოვარგოთ უნდა გამოვიყენოთ ბრძანება:

y = filter(B,A,x);

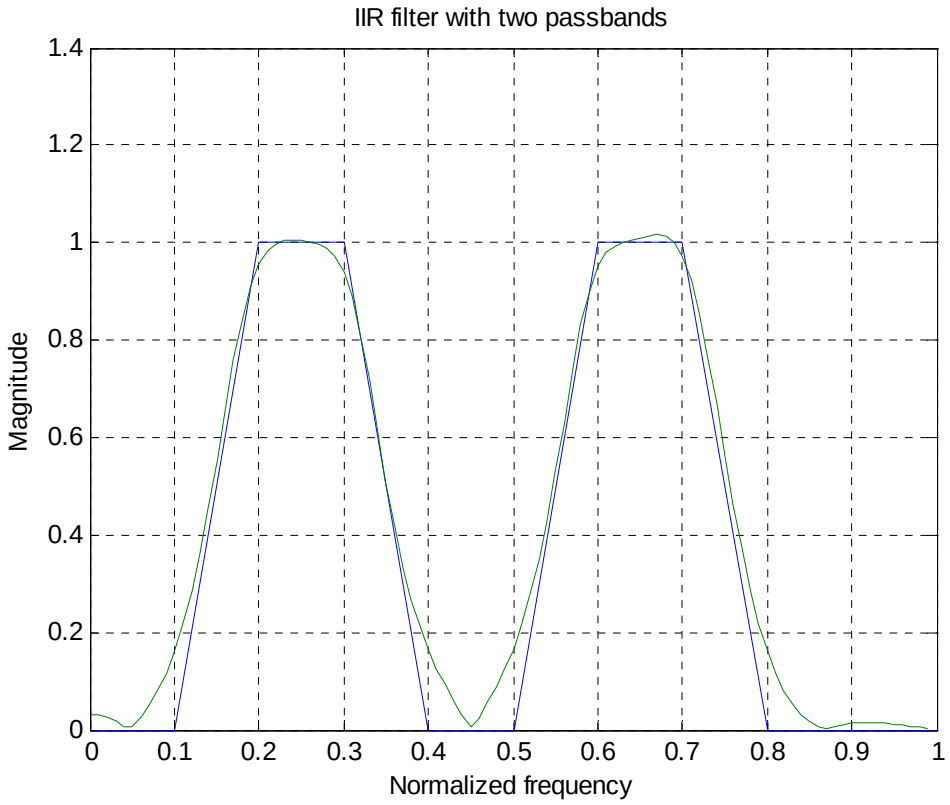
B და A მნიშვნელობები შეგვიძლია გამოვიყენოთ აგრეთვე ფილტრის სტანდარტული difference განტოლების განსასაზღვრავად.

პირდაპირი IIR ფილტრი

MATLAB-ს აქვს ფუნქცია Yule-Walker ფილტრისათვის. ვექტორები f და m ახასიათებს სიხშირულ-ამპლიტუდურ თვისებებს სიხშირის არეში 0-1.რაც შეესაბამება 0 დან ნაიქვისტის სიხშირემდე ინტერვალს. სიხშურეები f – ში უნდა დაიწყოს 0 – დან და დასრულდეს 1-ით.

დალაგებული უნდა იყოს ზრდის მიხედვით. ამპლიტუდები m–ში უნდა შეესაბამებოდეს სიხშირეებს f-ში. შემდეგი მაგალითში განხილულია ფილტრი გატარების 2 ზოლით;

```
m=[0 0 1 1 0 0 1 1 0 0];
f=[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 1];

[B,A] = yulewalk(12,f,m);
[H,wT] = freqz(B,A,100);
T = 0.001;

plot(f,m,wT/pi,abs(H)),...
title('IIR filter with two passbands'),...
xlabel('Normalized frequency'),ylabel('Magnitude'),grid
```



ნახ. 14.13 Yule-Walker ფილტრი

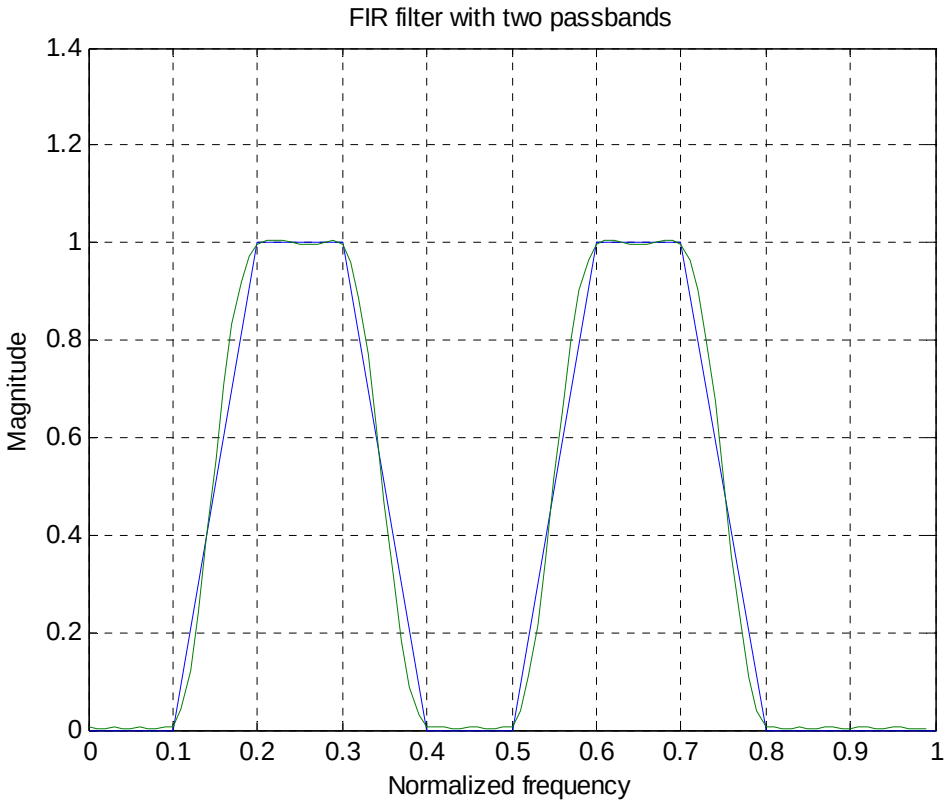
პირდაპირი FIR ფილტრი

ასეთი ფილტრი MATLAB-ში ხორციელდება Parks-MacClelanis ალგორითმით, რომელიც თავის მხრივ იყენებს **remez**-ის ალგორითმს. გავიხსენოთ, რომ FIR ფილტრს ესაჭიროება მხოლოდ B კოეფიციენტები, რადგან H(z) მნიშვნელის პოლინომი 1-ის ტოლია. MATLAB – ის ფუნქცია **remez** გვაძლევს ერთ ვექტორს :

```
B = remez(n,f,m)
```

პირველი არგუმენტია ფილტრის რიგი, f და m ძალიან გავს წინა თავში განხილულ სიდიდეებს, მაგრამ დამატებით წერტილების რაოდენიბა f და m ვექრორებში უნდა იყოს ლუწი.

```
m=[0 0 1 1 0 0 1 1 0 0];
f=[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 1];

[B,A] = remez(50,f,m);
[H,wT] = freqz(B,[1],100);


plot(f,m,wT/pi,abs(H)),…
title('FIR filter with two passbands'),…
xlabel('Normalized frequency'),ylabel('Magnitude'),grid
```



ნახ. 14.14 ფილტრი სასრული იმპულსური მახასიათებლით

სავარჯიშო

გამოყენეთ MATLAB –ის ფუნქციები შემდეგი სახის ფილტრების შესაქმნელად. ააგეთ ფილტრის მახასიათებელი მრუდი

1. დაბალსიხშირული IIR ფილტრი ზღვრული ცუტოფფ სიხშირით 75 ჰერცი, ათვლის სიხშირე 500 ჰერცი. (ფილტრის რიგი 5)

2. მაღალსიხშირული IIR ფილტრი ზღვრული cutoff სიხშირით 100 ჰერცი, ათვლის სიხშირე 1 კილოჰერცი. (ფილტრის რიგი 6)
3. დაბალსიხშირული FIR ფილტრი ზღვრული cutoff სიხშირით 75 ჰერცი, ათვლის სიხშირე 500 ჰერცი. (ფილტრის რიგი 5)
4. ზოლოვანი FIR ფილტრი გატარების ზოლით 100-200 ჰერცი, ათვლის სიხშირე 1 კილოჰერცი. (ფილტრის რიგი 80)

პრობლემა: არხების გამყოფი ფილტრი

კოსმოსური ხომალდის მიერ ღია სივრცეში გადაღებული გამოსახულება გადმოიცემა დედამიწაზე. ეს მონაცემები გარდაიქმნება ციფრულ სიგნალად, რომლის მიღების და დამუშავების შემდეგ მიიღება საწყისი გამოსახულება. ინფორმაციის მიღება წარმოებს რამდენიმე სენსორზე. სენსორზე მიღებული სიგნალის სიხშირული შედგენილობა დამოკიდებულია გაზომილი მონაცემების ტიპზე.

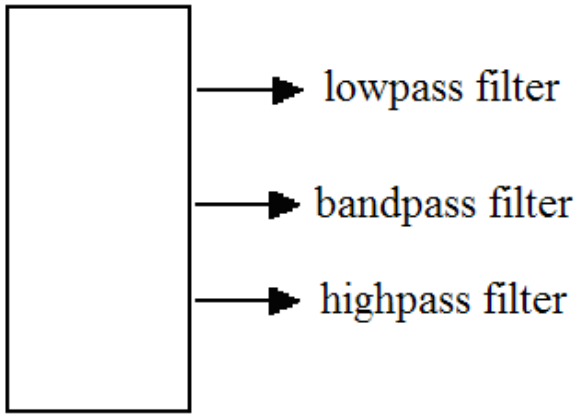
სხვადასხვა სიხშირის სიგნალები შეიძლება გაერთიანდეს ერთ სიგნალად მოდულაციის საშუალებით. მაგალითად დავუშვათ, გვინდა გადავცეთ 3 სიგნალი პარალელურად. პირველი სიგნალი შეიცავს სიხშირულ კომპონენტებსს 0-დან 100 ჰერცამდე, მეორე-500 ჰერციდან 1 კილოჰერცამდე, მესამე კი 2 კილოჰერციდან 5 კილოჰერცამდე. დავუშვათ სიგნალი რომელიც შეიცავს ამ სამ სიგნალს აითვლება სიხშირით 10 კილოჰერცი. იმისათვის, რომ მიღებული სიგნალიდან გამოვყოთ თითოეული მდგენელი ცალკ-ცალკე, გვჭირდება დაბალსიხშირული ფილტრი ზედა ზღვრული სიხშირით 100 ჰერცი, ზოლოვანი ფილტრი სიხშირეთა ინტერვალით 500-1000 ჰერცი და მაღალსიხშირული ფილტრი ქვედა ზღვრული სიხშირით 2 კილოჰერცი. ფილტრის რიგი საკმაოდ მაღალი უნდა იყოს, რომ მივიღოთ ვიწრო გარდამავალი ზოლი რათა სიხშირეები რაც შეიძლება მკვეთრად გაიმიჯნოს.

1. ამოცანის დასმა:

შევქმნათ 3 ფილტრი, რომელიც უნდა გამოვიყენოთ 10 კილოჰერც სიხშირით ათვლილი სიგნალისათვის. ერთი მათგანი უნდა იყოს დაბალსიხშირული ფილტრი, რომელიც ჩამოჭრის 100 ჰერცზე უფრო მაღალ სიხშირულ კომპონენტებს, მეორე უნდა იყოს ზოლური ფილტრი, რომელიც გაატარებს სიხშირეებს 500-1000 ჰერც ინტერვალში, მესამე ფილტრი უნდა იყოს მაღალსიხშირული, რომელიც გაატარებს ყველა სიხშირეს 2 კილოჰერციდან.

2. INPUT/OUTPUT აღწერა

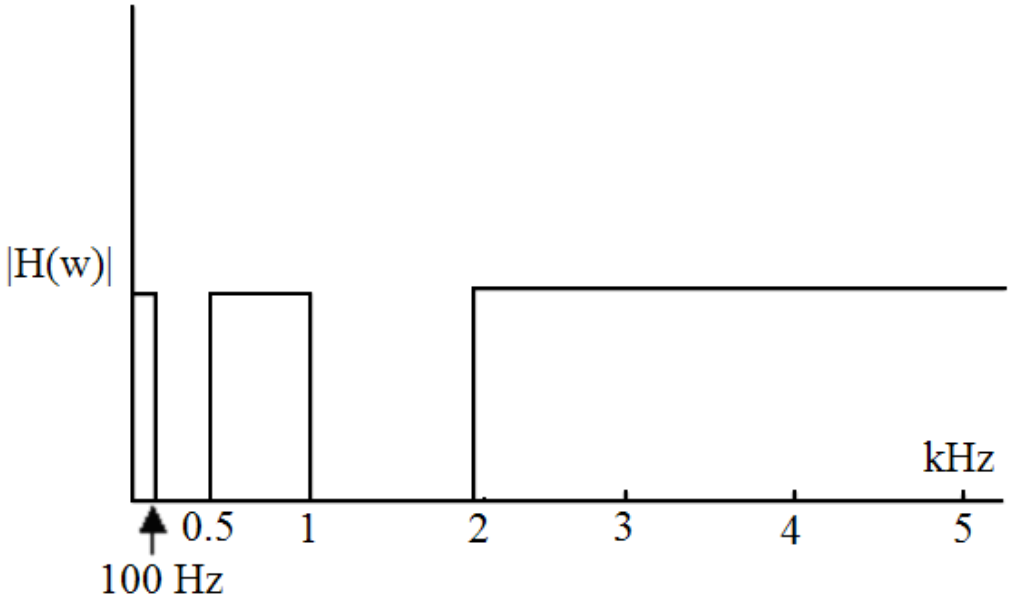
ამ პრობლემაში არ გვაქვს საწყისი მონაცემები, ხოლო შედეგად ვრებულობთ ვექტორებს კოეფიციენტებით, რომელიც განსაზღვრავს ამ სამი ფილტრის გადაცემის ფუნქციებს. იხ. დიაგრამა ნახ. 14.15.



ნახ. 14.15 I/O დიაგრამა

3. სახელდახელო ამოხსნა

ნახ. 14.17 ნაჩვენებია სიხშირული ინტერვალი 0-დან ნაიქვისტის სიხშირემდე (500 კჰც) სამი ფილტრითურთ. უნდა გამოვიყენოთ უტტერწორტჰ ფილტრი იმისათვის, რომ მივიღოთ გატარების და მოკვეთის ბრტყელი ზოლი. შესაძლოა დაგვჭირდეს ექსპერიმენტი ფილტრის რიგის შესარჩევად, იმისათვის რომ დავრწმუნდეთ ფილტრების გატარების ზოლები არ ფარავენ ერთმანეთს და ამდენად არხები მკვეთრად გაიმიჯნება.



ნახ. 14.16 ფილტრის ესკიზი

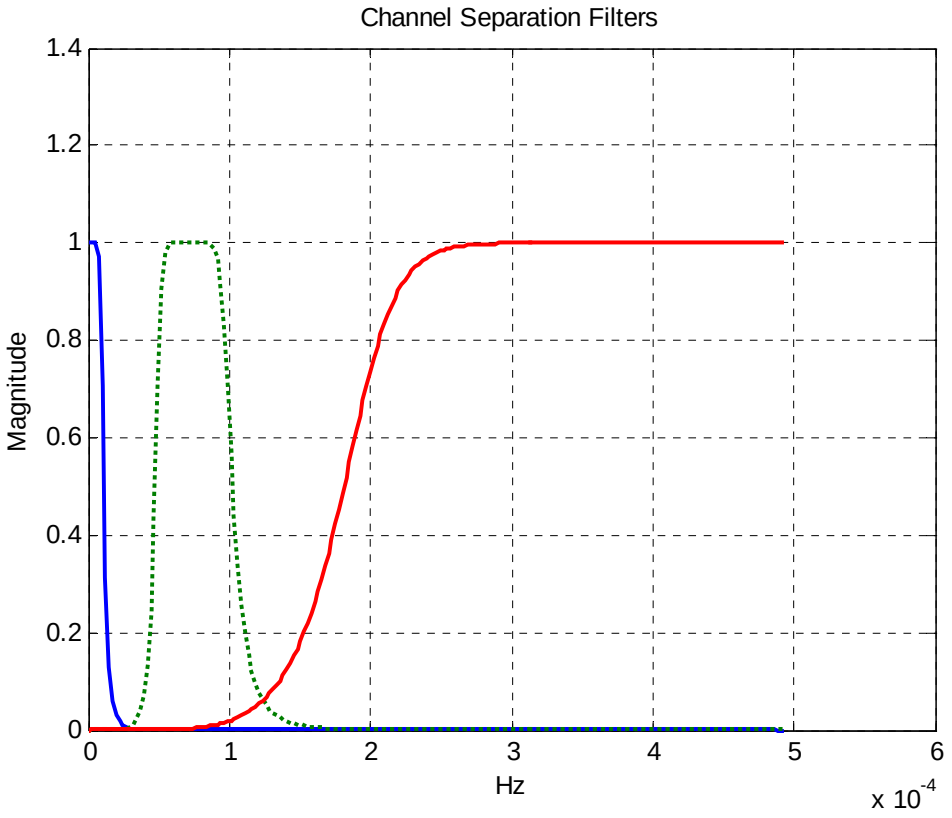
4. MATLAB ამოხსნა

ეს პროგრამა ითვლის ნორმირებულ სიხშირეებს(0-დან 1-მდე 1-შეესაბამება ნაიქვისტის სიხშირეს) ზღვრული სიხშირეებისათვის **butter** ფუნქციაში, მას შემდეგ რაც გამოვითვლით კოეფიციენტებს, მივმართვთ **freqz** ფუნქციას, რომ ავაგოთ ფილტრის მახასიათებელი მრუდი. გავიხსენოთ, რომ freqz ფუნქცია ახდენს სიხშირეთა ნორმირებას 0 - π ინტერვალში, სადაც π წარმოადგენს ნაიქვისტის სიხშირეს. სიხშირის ერთეულად ავიღებთ ჰერცს.

```
%         This program designs three digital filters
%         for use in a channel separation problem

fs = 10000;      %sampling frequency

T= 1/fs;         %sampling time
fn = fs/2;       %Nyquist frequency
f1n = 100/fn;    %normalized lowpass cutoff
f2n = 500/fn;    %normalized bandpass left cutoff
f3n = 1000/fn;   %normalized bandpass left cutoff
f4n = 2000/fn;   %normalized highpass cutoff
%
[B1,A1] = butter(5,f1n);
[B2,A2] = butter(5,[f2n,f3n]);
[B3,A3] = butter(5,f4n,'high');
%
[H1,wT] = freqz(B1,A1,200);
[H2,wT] = freqz(B2,A2,200);
[H3,wT] = freqz(B3,A3,200);
%
hertz = wT/2*pi*T;
plot(hertz,abs(H1),'-
',hertz,abs(H2),':',hertz,abs(H3),'r'),...
    title('Channel Separation Filters'),...
    xlabel('Hz'),ylabel('Magnitude'),grid
```

5. შემოწმება

სამივე ფილტრის მახასიათებელი მრუდი (მოდული) წარმოდგენილია



ნახ. 14.17 არხების გამყოფი სამი ფილტრის მახასიათებელი მრუდები

განხილული იქნა MATLAB-ის რამდენიმე ფუნქცია, რომელიც საჭიროა სიგნალის დამუშავებისათვის. ციფრული სიგნალის სიხშირული შედგენილობის ანალიზისათვის გამოვიყენეთ **fft** ფუნქცია. ანალოგური და ციფრული ფილტრების სიხშირული მახასიათებლების შესასწავლად გამოვიყენეთ **freqs** და **freqz** ფუნქციები. კომპლექსური სიგნალიდან შეგვიძლია გამოვითვალოთ ფილტრის ამპლიტუდა და ფაზა. **filter** ფუნქცია გამოიყენება **IIR** და **FIR** ფილტრების განსახორციელებლად. და ბოლოს განხილული იქნა რამდენიმე ფუქცია **IIR** და **FIR** ფილტრების დიზაინისთვის.

MATLAB ბრძანებები და ფუნქციები

butter	Butterworth ციფრული ფილტრის დიზაინი
cheby1	Chebyshev I ტიპის ფილტრის დიზაინი
cheby2	Chebyshev II ტიპის ფილტრის დიზაინი
ellip	ელიფსური ციფრული ფილტრის დიზაინი
fft	გამოითვლის სიგნალის სიხშირულ შემცველობას (სწრაფი ფურიე გარდაქმნა)
filter	შემომავალ სიგნალს მოარგებს ციფრულ ფილტრს
freqs	გამოითვლის ანალოგურ სიხშირულ შემცველობას
freqz	გამოითვლის ციფრულ სიხშირულ შემცველობას
grpdelay	დაადგენს ციფრული სიგნალის ჯგუფური შეყოვნების ზომას
ifft	გამოითვლის FFT შებრუნებულს (შებრუნებული ფურიე გარდაქმნა)
remez	ოპტიმალური FIR ციფრული ფილტრის დიზაინი
residue	რაციონალურ წილადად დაშლა

unwrap	გვაძლევს ფაზურ კუთხის ყველა შესაძლო მნიშვნელობას
yulewalk	ოპტიმალური IIR ციფრული ფილტრის დიზაინი

პრობლემები

1-6 ამოცანები დაკავშირებულია ამ თავში განხილულ პრობლემებთან, ხოლო 7-23 – ახალ საინჟინრო პრობლემებს უკავშირდება

არხების გამყოფი ფილტრი. ეს პრობლემა უკავშირდება ამ თავში განხილულ პრობემას არხების გამყოფი ფილტრის დიზაინის შესახებ. შევეცადოთ შევქმნათ სისტემის კომპიუტერული მოდელი:

1. პირველ რიგში უნდა შევქმნათ სიგნალი, რომელიც შეიცავს სხვადასხვა სიხშირულ მდგენელს. ამას მივაღწევთ სხვადასხვა სიხშირის და ამპლიტუდეს სინუსოიდათა შეკრებით, რომლებიც აითვლება 10 კილოჰერცი სიხშირით. პირველი სიგნალი შედგება სამი სინუსოიდისაგა, რომელთა სიხშირეებია: 25 ჰც, 40ჰც და 75ჰც, მეორე სინუსოიდა შეიცავს სამ სინუსოიდას სიხშირეებით: 500ჰც, 730 ჰც და 850ჰც, მესამე სინუსოიდა კი წარმოადგებს სამი სინუსოიდის ჯამს მაღალი სიხშირეებით: 3 500ჰც, 4 000ჰც და 4 200 ჰც. შეარჩიეთ სინუსოიდებისთვის სხვადასხვა ამპლიტუდები და ფაზები და ააგეთ 500 წერტილი პირველი, მეორე და მესამე სიგნალისათვის გრაფიკი ცალკ-ცალკე.
2. გამოთვლეთ და ააგეთ პირველ ამოცანაში აღწერილი სიგნალების სიხშირული მახასიათებლები (მაგნიტუდა და ფაზა). გრაფიკის აგებისას x ღერძზე დაიტანეთ სიხშირის მნიშვნელობები ჰერცებში.
3. შეკრიბეთ პირველ ამოცანაში აღწერილი სიგნალები. ააგეთ სიგნალი დროით და სიხშირულ სივრცეში(არეში). x ღერძზე გადაზომეთ სიხშირე ჰერცებში.
4. მოარგეთ დაბალსიხშირული ფილტრი მესამე ამოცანაში აღწერილ სიგნალს. ააგეთ შედეგად მიღებული სიგნალის დროითი და სიხშირული (მაგნიტუდე) მახასიათებლების გრაფიკი. შეადარეთ იგი 1 და 2 ამოცანაში აღწერილ სათანადო სიგნალის გრაფიკს. დროითი გრაფიკი პირველი სიგნალისათვის მისი მსგავსი უნდა იყოს შესაძლო ფაზური წანაცვლების გარეშე, ხოლო სიხშირული მრუდები თითქმის იდენტური უნდა იყოს.
5. გაიმეორეთ მე-4 ამოცანა ზოლოვანი ფილტრისათვის. შეადარეთ იგი 1 და 2 ამოცანაში აღწერილ სათანადო სიგნალის გრაფიკს. დროითი გრაფიკი მეორე სიგნალისათვის მისი მსგავსი უნდა იყოს შესაძლო ფაზური წანაცვლების გარეშე, ხოლო სიხშირული მრუდები თითქმის იდენტური უნდა იყოს.
6. გაიმეორეთ მე-4 ამოცანა მაღალსიხშირული ფილტრისათვის. შეადარეთ იგი 1 და 2 ამოცანაში აღწერილ სათანადო სიგნალის გრაფიკს. დროითი გრაფიკი მესამე სიგნალისათვის მისი მსგავსი უნდა იყოს შესაძლო ფაზური წანაცვლების გარეშე, ხოლო სიხშირული მრუდები თითქმის იდენტური უნდა იყოს.

ფილტრის მახასიათებლები. თითოეული ფილტრისათვის განსაზღვრეთ მახასიათებლები ზღვრული, გარდამავალი და მოკვეთის (stopband). ზღვრული სიხშირის შესაფასებლად გამოიყენეთ 0.7 დონე, ხოლო მოკვეთის სიხშირისათვის 0.1.

7. $H(s) = \dfrac{0.5279}{s^2 + 1.0275s + 0.5279}$

8. $H(s) = \dfrac{s^2}{s^2 + 0.1117s + 0.0062}$
9. $H(s) = \dfrac{1.05s}{s^2 + 1.05s + 0.447}$
10. $H(s) = \dfrac{s^2 + 2.2359}{s^2 + 2.3511s + 2.2359}$
11. $H(z) = \dfrac{0.2066 - 0.4131z^{-1} + 0.2066z^{-2}}{1 - 0.3695z^{-1} + 0.1958z^{-2}}$
12. $H(z) = \dfrac{0.894 - 1.789^{-1} + 0.894z^{-2}}{1 - 1.778z^{-1} + 0.799z^{-2}}$
13. $H(z) = \dfrac{0.42 - 0.42z^{-2}}{1 - 0.443z^{-1} + 0.159z^{-2}}$
14. $H(z) = \dfrac{0.5792 - 0.4425z^{-1} + 0.1584z^{-2}}{1 - 0.4425z^{-1} + 0.1584z^{-2}}$
15. $y_n = 0.04x_{n-1} + 0.17x_{n-2} + 0.25x_{n-3} + 0.17x_{n-4} + 0.04x_{n-5}$
16. $y_n = 0.42x_n - 0.42x_{n-2} + 0.44y_{n-1} - 0.16y_{n-2}$
17. $y_n = 0.33_{n+1} + 0.33x_n + 0.33x_{n-1}$
18. $y_n = 0.33x_n + 0.33x_{n-1} + 0.33x_{n-2}$

19. შექმენით დაბალსიხშირული ფილტრი ზღვრული სიხშირით 1 კილოჰერცი, როცა ათლის სიხშირეა 8 კილოჰერცი. შეადარე ერთმანეთს 4 სტანდარტული IIR ფილტრის მახასიათებლები მე-8 რიგის შესაბამის ფილტრის მახასიათებლებს ერთიდაიგივე ნახაზზე.
20. შექმენით მაღალსიხშირული ფილტრი ზღვრული სიხშირით 500 ჰერცი, როცა ათლის სიხშირეა 1500 ჰერცი. შეადარე ერთმანეთს მე-8 რიგის ელიფსური და 32-ე რიგის FIR ფილტრის მახასიათებლები ერთიდაიგივე ნახაზზე.
21. შექმენით ზოლოვანი ფილტრი გატარების სიხშირით 300-4000 ჰერცი, როცა ათლის სიხშირეა 9.6 კილოჰერცი. შეადარე ერთმანეთს Butterworth და მე-8 მრიგის FIR ფილტრის მახასიათებლები ერთიდაიგივე ნახაზზე.
22. შექმენით ფილტრირომელიც არ გაატარებს სიხშირულ ზოლს 500-100 ჰც, როცა ათლის სიხშირეა 10 კილოჰერცი. შეადარე ერთმანეთს 12-ე რიგის ელიფსური და მე-12 რიგის ულე-ჭალკერ ფილტრის მახასიათებლები ერთიდაიგივე ნახაზზე.
23. შექმენით ფილტრი, რომელიც გამორიცხავს სიხშირეთა ზოლს 100-150 ჰც და 500-600ჰც, როცა ათლის სიხშირეა 2.5 კილოჰერცი. შეადარე ერთმანეთს IIR და FIR ფილტრის მახასიათებლები ერთიდაიგივე ნახაზზე.

15 გამონასახის დამუშავება

MATLAB გააჩნია სტანდარტული ალგორითმებისა და გრაფიკულ საშუალებათა ერთობლიობა გამონასახის დამუშავებისათვის. შესაძლებელი დაბალი გარჩევის, დაზიანებული გამონასახის აღდგენა და გაუმჯობესება. ამოვჭრათ დეტალები გამონასახიდან და გავაანალიზოთ გამოსახულება ფორმისა თუ სტრუქტურის თვალსაზრისით. Tulbox- Image Processing – გვთავაზობს ამისათვის უკვე დაწერილ M-ფუნქციებს, თუმცა მომხმარებელს შეუძლია თავად შექმნას საჭირო ალგორითმის მიხედვით ახალი **MATLAB** ფუნქციები.

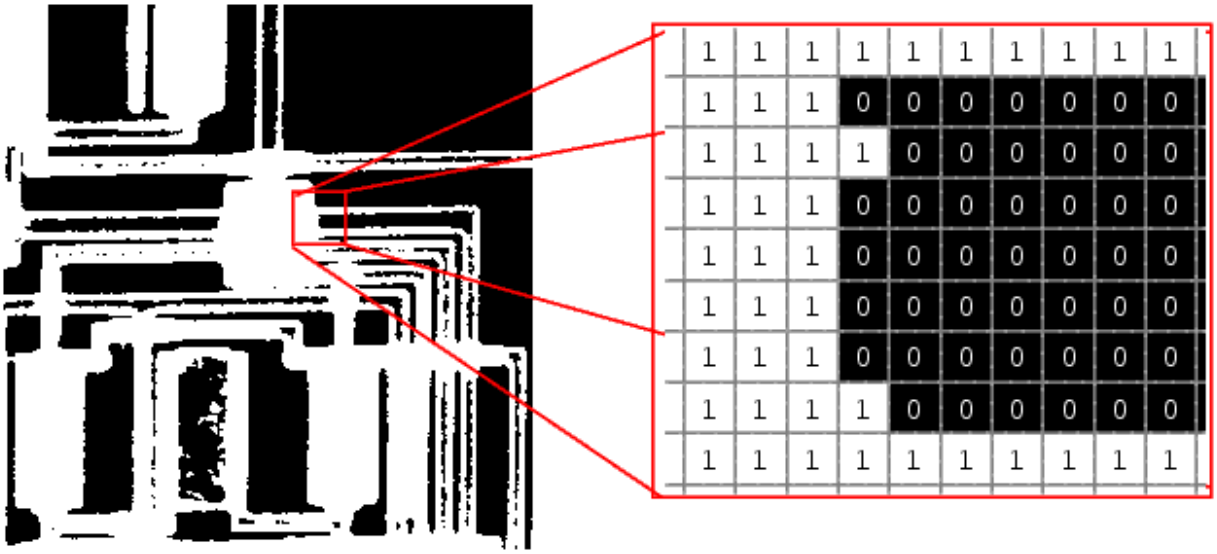
MATLAB საშუალებას გვაძლევს დავამუშაოთ გამონასახები, რომელნიც მიღებულია სხვადახვა ტიპის მიმღებებით როგორიცაა ციფრული კამერა, ფრეიმ-გრაბერი, კოსმოსურ თანამგზავრებზე დადგმული სხვადასხვა ტიპის მიმღები, მიმღები, რომლითაც აღჭურვილია საედიცინო მიკროსკოპი, ტელესკოპი თუ სხვა მრავალი სამეცნიერო ინსტრუმენტი.

15.7 ბრძანებები და ფუნქციები

15.1 შესავალი

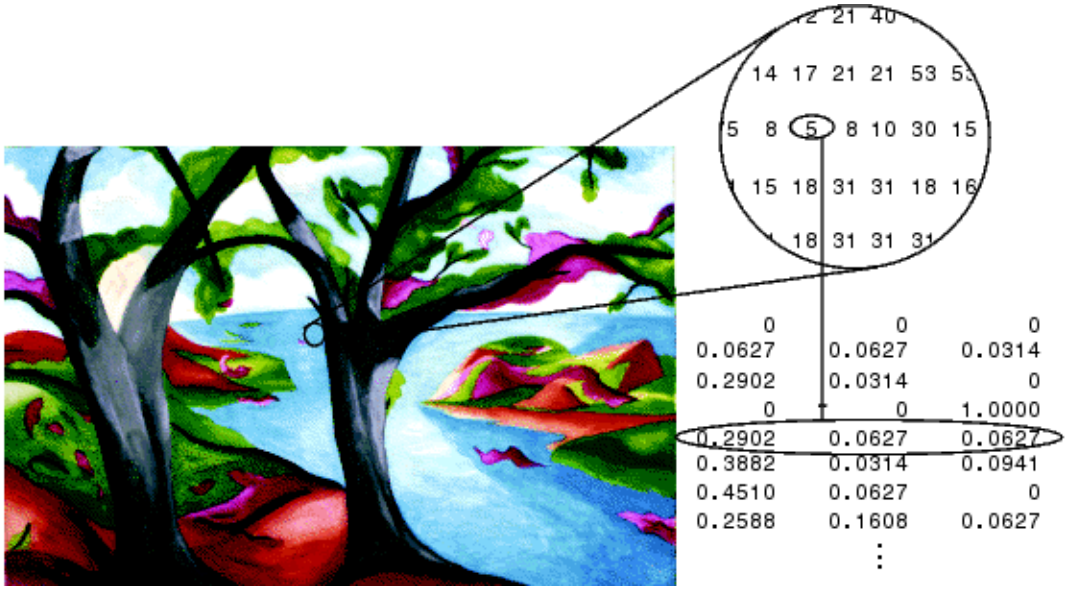
MATLAB გააჩნია სპეციალური ფუნქციათა კრებული გამონასახისათვის. იმისათვის რომ შევამოწმოთ არის თუ არა სისტემაში ინსტალირებული **Image** Processing Toolbox ვსარგებლობთ ფუნქციით **ver** ფუნქცია გვაწვდის ინფორმაციას MATLAB და Toolbox შესახებ, რომლებიც ინსტალირებულია მოცემულ სისტემაში.
გამონასახის 4 ძირითადი ტიპი არსებობს:

- ბინარული, ორობითი გამონასახი – ლოგიკური მასივი რომელიც შეიცავს მხოლოდ 0 და 1 მნიშვნელობებს.



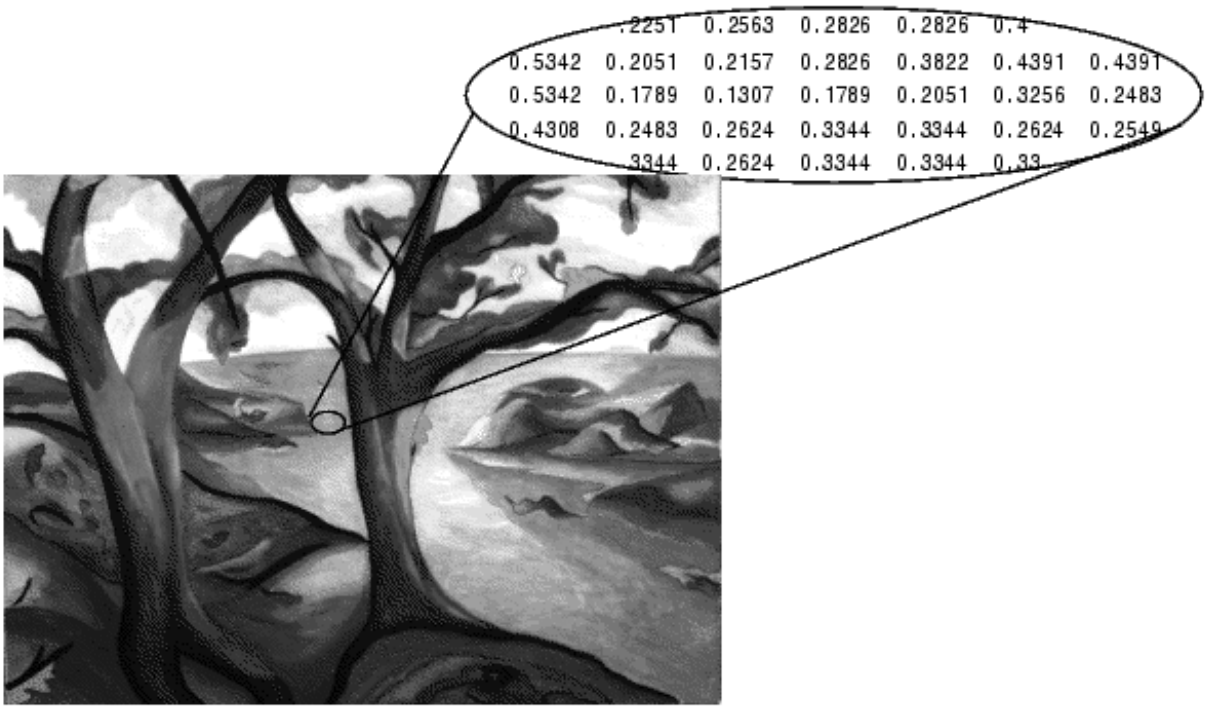
ნახ. 15.1 ორობითი გამონასახი

- ინდექსირებული – მასივი რომელიც შესაძლოა იყოს ლოგიკური, uint8, uint16, single ან double ტიპის. რომლის პიქსელების მნიშვნელობა არის პირდაპირ color map ინდექსები, ხოლო colormap არის m X 3 double ტიპის მატრიცა ინფორმაციით ფერების შესახებ. მას სხვაგვარად ფსევდოფერად გამონასახსაც უწოდებენ.



ნახ. 15.2 ინდექსირებული გამონასახი

- შავ-თეთრი - uint8, uint16, single ან double ტიპის მატრიცა, რომლის პიქსელებიც წარმოდგენილია ინტენსივობებში. single ან double ტიპისათვის პიქსელის მნიშვნელობებია [0,1], uint8 - [0,255], uint16 – [0, 65535].



ნახ. 15.3 ინდექსირებული გამონასახი

- რეალური (truecolor) – სამგანზომილებიანი მატრიცა uint8, uint16, single ან double ტიპის, რომლის პიქსელებიც წარმოდგენილია ინტენსივობებში. ასევე უწოდებენ RGB გამონასახს. single ან double ტიპისათვის პიქსელის მნიშვნელობებია [0,1], uint8 - [0,255], uint16 – [0.65535]

ნახ. 15.4 RGB გამონასახი

15.2 გარდაქმნა გამონასახის ტიპებს შორის

შემდეგი ფუნქციები გამოიყენება გამონასახის ერთი ტიპის სხვა ტიპის გამონასახად გარდასაქმნელად:

dither – შავ-თეთრი გამონასახი ბინარულში, ან რეალური - ინდექსირებულში

gray2ind – შავ-თეთრი - ინდექსირებულში

im2bw – შავ-თეთრი, ინდექსირებული ან რეალური გამონასახი ორობით გამონასახში

ind2gray – ინდექსირებული შავ-თეთრში

ind2grb – ინდექსირებული რეალურ გამონასაში (MATLAB 9.0 version)

mat2gray – მონაცემთა მატრიცა გადაყავს შავ-თეთრ გამონაახში მონაცენთა სკალირების საფუძველზე

rgb2gray– რეალური –შავ-თეთრში

rgb2ind – რეალური – ინდექსირებულში

15.3 გამონასახის იმპირტ-ექსპორტი

გამონასახის იმპორტისა და ექსპორტისათვის **MATLAB** გარემოში, რამდენიმე გზა არსებობს.

- **Tulbox – Image Ecquisition**, რომელიც საშუალებას იძლევა პირდაპირ მიმღებიდან **MATLAB** გარემოში მივიღოთ გამოსახულება.
- **Tulbox – Database** შექმნილია ODBC/JDBC ფორმატის მონაცემთა ბაზაზე სამუშაოდ
- შესაძლებელია სტანდარტული ფორმატის გამონასახზე მუშაობა, როგორიცაა: JPEG, TIFF, PNG, HDF, HDF-EOS, FITS, Microsoft Excel, ASCII და ბინარული ფაილები.

გამონასახის გამოძახება **MATLAB** გარემოში ხორციელდება ფუნქციის **imread**, გრაფიკულ ფანჯარაში მისი წარმოდგენა – **imshow.**

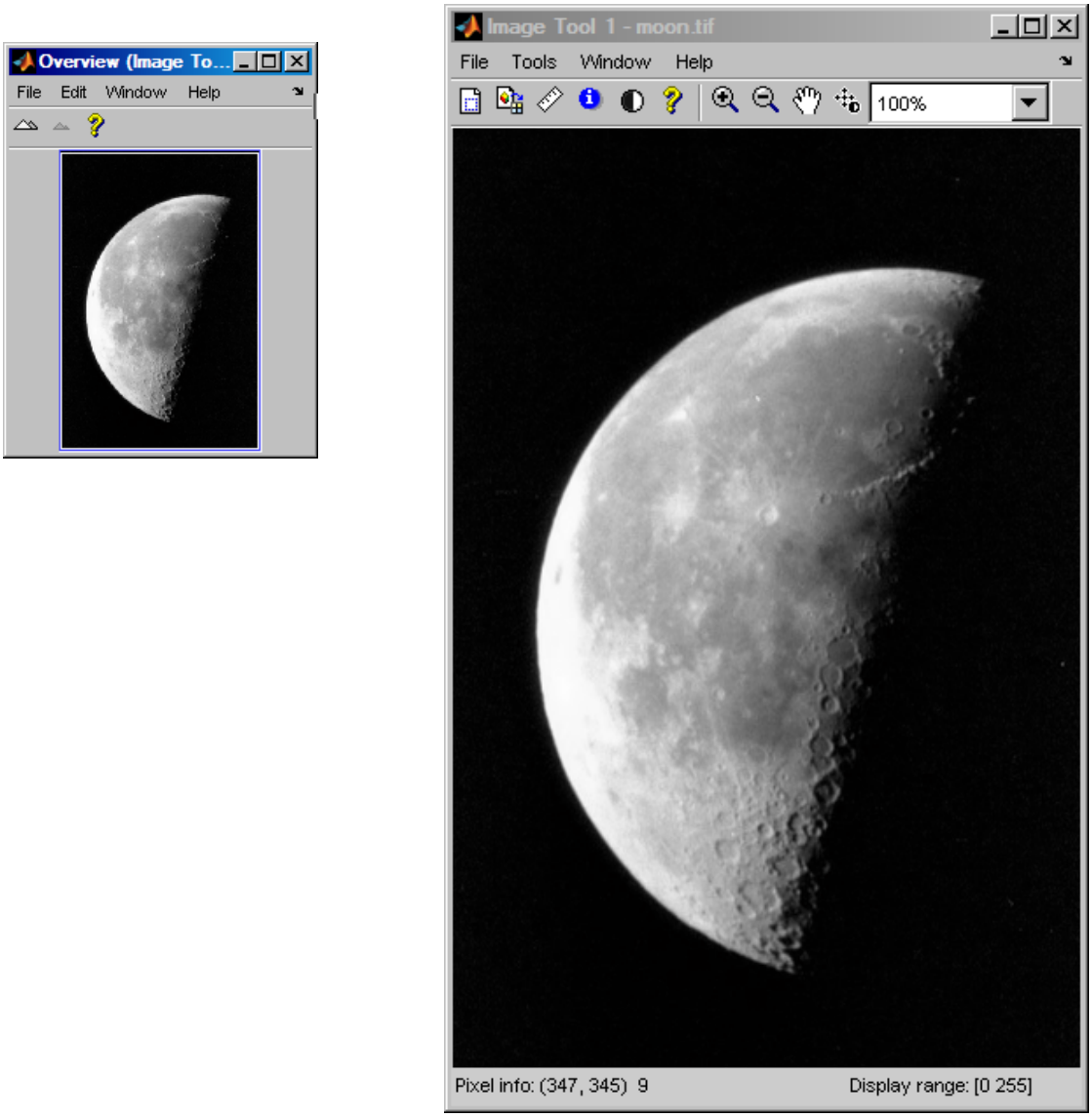
A=imread(‘filename’)	გამოიძახებს ორფეროვანი(შავ-თეთრი) ან ფერად გამონასახის შემცველ ფაილს და წარმოადგენს მას A მატრიცის სახით. თუ გამონასახი შავ-თეთრია, მივიღებთ ორგანზომილებიან მატრიცას, თუ ფერადია, მივიღებთ სამგანზომილებიან მატრიცას. მატრიცის კლასი დამოკიდებულია გამოძახებული ფაილის ფორმატზე.
[X, map]=imread(‘filename’)	გამოიძახებს ფერადი, ინდეხსირებული გამონასახის შემცველ ფაილს და წარმოადგენს მას X მატრიცის სახით, რომლის შესაბამის ინფორმაციას ფერთა შესახებ წარმოდგენილია map მატრიცის სახით
imshow(‘filename’)	წარმოადგენს გამონასახის შემცველ ფაილს გრაფიკულ ფანჯარაში

იმისათვის, რომ ვნახოთ თუ როგორი სახით წარმოადგენს გამონასახს ფუნქცია imread მივმართოთ ფუნქციას whos:

```
imread('moon.tif');
whos
  Name      Size                    Bytes  Class
  ans     537x358                  192246  uint8 array
Grand total is 192246 elements using 192246 bytes
```

- ფუნქცია **imtool** გამონასახის გრაფიკულ ფანჯარაში წარმოდგენასთან ერთად საშუალებას გვაძლევს გამოვიყენოთ ასევე Pixel Region tool, the Image Information tool, და the Adjust Contrast tool:

```
imtool('moon.tif')
```



ნახ. 15.5 imtool

15.4 არითმეტიკული ოპერაციები გამონასახზე

მას შემდეგ, რაც გამონასახი გამოძახებულ იქნება **MATLAB** გარემოში ბრძანებით **imread**, შესაძლებელია მიღებულ ინდექსირებულ გამონასახზე არითმეტიკული ოპერაციების წარმოება. გამონასახები ერთიდაიგივე ზომის და კლასის უნდა იყოს. მათემატიკური ოპერაციები სრულდება შესაბამის ელემენტებზე.

imadd	გამონასახების შეკრება
imdevide	გამონასახების გაყოფა
imsubtract	გამონასახების გამოკლება
immultiply	გამონასახების გამრავლება
imabsdiff	გვაძლევს ახალ მატრიცას, რომლის ელემენტებიც არგუმენტთა შესაბამისი ელემენტების სხვაობათა

აბსოლუტურ სიდიდეს წარმოადგენს

ამ ფუნქციათა არგუმენტები ერთნაირი ზომის და კლასის მატრიცებია, ან მეორე ელემენტი შესაძლოა იყოს double scalar კლასის.
მაგალითად:

```
I = imread('rice.png');
J = imread('cameraman.tif');
K = imadd(I,J);
figure, imshow(K)
```

ან:

```
I = imread('moon.tif');
J = immultiply(I,0.5);
figure, imshow(I), figure, imshow(J)
```

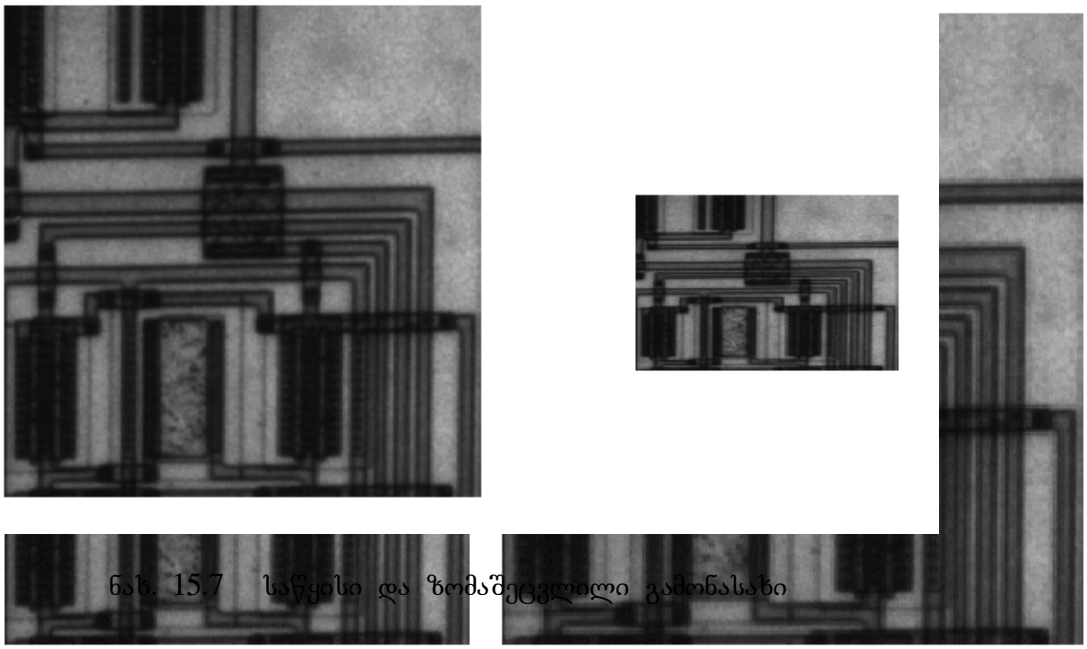
15.5 გამონასახის ზომის შეცვლა

MATLAB საშუალებით შესაძლებელია გამონასახის ზომის შეცვლა ფუნქციის **imresize**. განვიხილოთ ამ ფუნქციის გამოყენების რამდენიმე მაგალითი.

- გამონასახის ზომის შეცვლა გადიდების ფაქტორის მითითებით

```
I = imread('circuit.tif');
J = imresize(I,1.25);
imshow(I)
figure, imshow(J)
```

ბრძანებათა ეს მწკრივი გამოიძახებს MATLAB გარემოში გამონასახს და მოგვცემს საწყისი გამონასახის ინდექსირებულ მატრიცას I, 1.25 ჯერ გადიდებული ახალი გამონასახის ინდექსირებულ მატრიცას J, საწყისი და ახალი გამონასახის გრაფიკულ გამოსახულებებს ნახ. 15.6.



ნახ. 15.7 საწყისი და ზომაშეცვლილი გამონასახი

ნახ. 15.6 საწყისი და გადიდებული გამონასახი

- გამონასახის ზომის შეცვლა სასურველი ზომის მითითებით

```
I = imread('circuit.tif');
J = imresize(I,[100 150]);
imshow(I)
figure, imshow(J)
```

მივიღებთ საწყის და ზომაშეცვლილ გამონასახებს ნახ. 15.7

- გამონასახის ზომის შეცვლა ინტერპოლირების მეთოდის მითითებით

```
I = imread('circuit.tif');
J = imresize(I,[100 300],'bilinear');
imshow(I)
figure, imshow(J)
```

ფუნქციის მესამე არგუმენტი მიუთითებს ინტერპოლირების მეთოდს, რომელიც უნდა იქნას გამოყენებული ახალი გამონასახის შესაქმნელად. თუ არგუმენტი მითითებული არ არის ინტერპოლირება ხდება მეზობელი ელემენტების მეთოდით, `'bilinear'` - ინტერპოლირება ორი წრფივი ფუნქციის გამოყენებით, `'bicubic'`- ინტერპოლირება ორი კუბური ფუნქციის გამოყენებით.

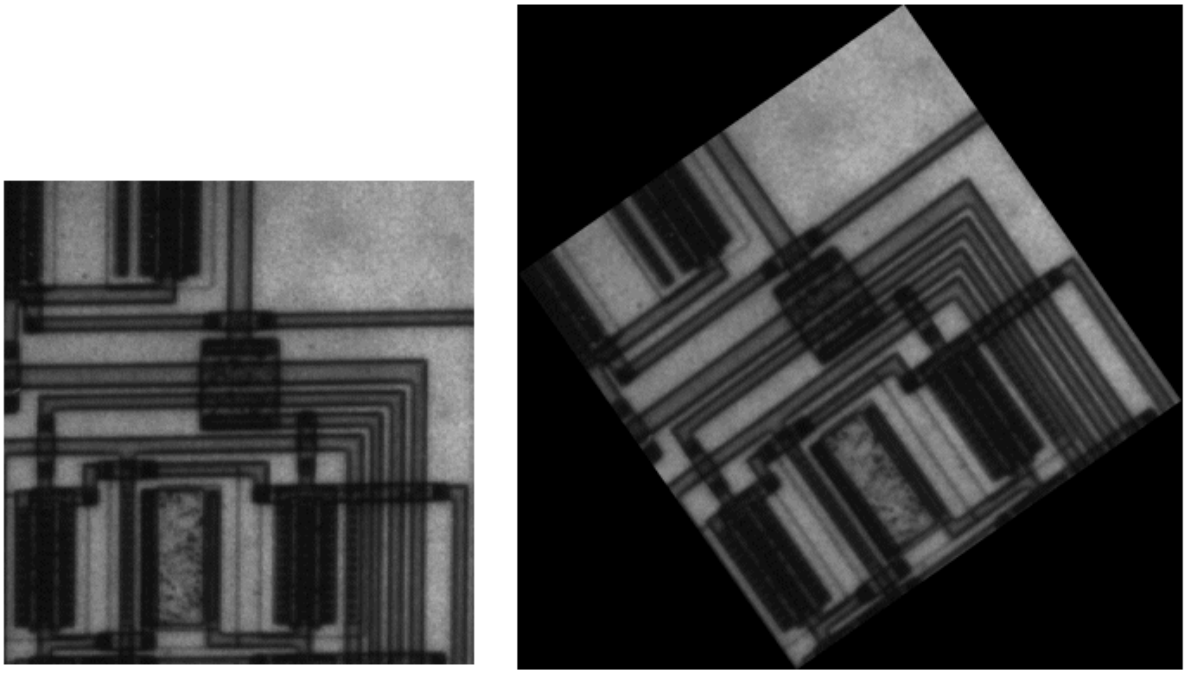
15.6 გამონასახის მობრუნება

MATLAB საშუალებით სესაძლებელია გამონასახის მობრუნება ფუნქციის imrotate. განვიხილოთ ამ ფუნქციის გამოყენების მაგალითი.

- გამონასახის მობრუნება

```
I = imread('circuit.tif');
J = imrotate(I,35,'bilinear');
imshow(I)
figure, imshow(J)
```

ბრძანებათა ეს მწკრივი გამოიძახებს MATLAB გარემოში გამონასახს და მოგვცემს საწყისი გამონასახის ინდექსირებულ მატრიცას I, 35 გრადუსით მობრუნებული ახალი გამონასახის ინდექსირებულ მატრიცას J, საწყისი და ახალი გამონასახის გრაფიკულ გამოსახულებებს ნახ. 15.8.



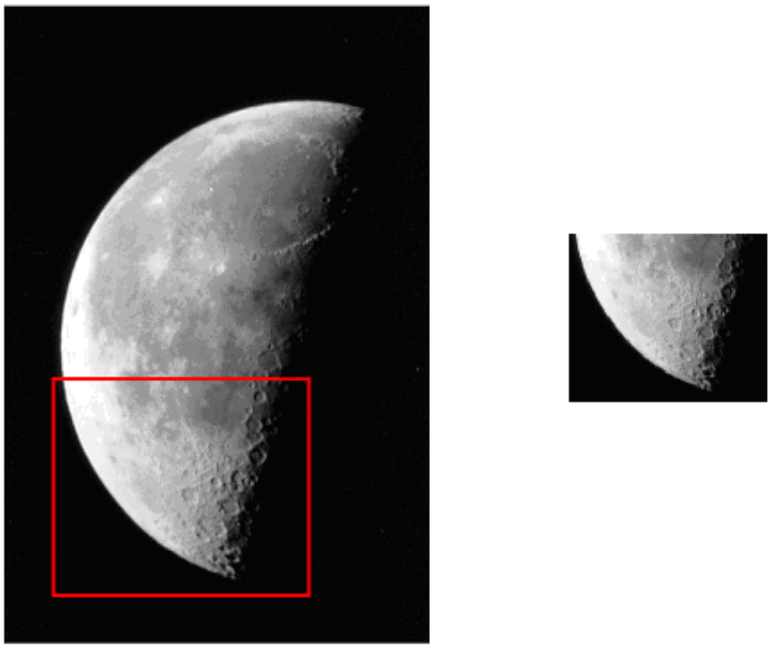
ნახ. 15.8 საწყისი და შემობრუნებული გამონასახი

15.7 დეტალის ამოჭრა გამონასახიდან

MATLAB-ში დეტალის ამოსაჭრელად გამოიყენება ფუნქცია imcrop. განვიხილოთ ამ ფუნქციის გამოყენების მაგალითი.

```
imshow moon.tif
I = imcrop;
imshow(I);
```

ფუნქცია imcrop ეკრანზე მოიხმობს გამონასახს და დაელოდება მომხმარებელს მაუზის საშუალებით მონიშნოს სასურველი დეტალი გამონასახიდან. მივიღებთ ამოკვეთილი დეტალის გრაფიკულ გამოსახულებას ნახ. 15.9.



ნახ. 15.9 საწყისი გამონასახი და ამოკვეთილი დეტალი

15.8 ინფორმაცია გამონასახის შესახებ და სტატისტიკური ანალიზი

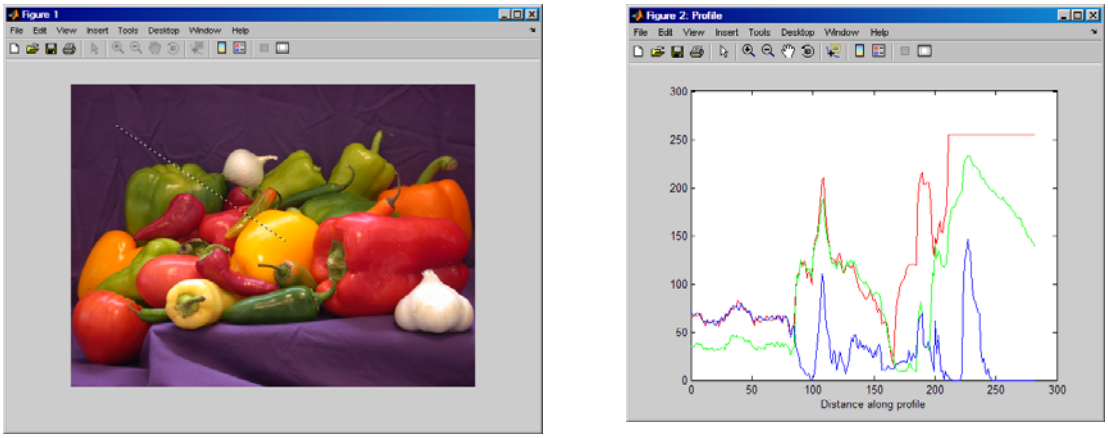
MATLAB გვაძლევს საშუალებას მივიღოთ ინფორმაცია იმ სიდიდეთა შესახებ, რაც ქმნის გამონასახს და შევასრულოთ ამ მონაცემთა სტატისტიკური ანალიზი.

- ინფორმაცია გამონასახის შემადგენელი პიქსელის შესახებ. ფუნქცია **impixel** იმისათვის, რომ მივიღოთ ინფორმაცია პიქსელის შესახებ, უნდა გამოვიძახოთ გამონასახი imshow canoe.tif. ამის შემდეგ ვისარგებლოთ ბრძანებით: vals = impixel, რაც საშუალებას მოგვცემს გამონასახზე მოვნიშნოთ ჩვენთვის საინტერესო პიქსელები. შერჩევის პროცესი სრულდება return კლავიშის საშუალებით. ინფორმაცია მონიშნული პიქსელების შესახებ შეინახება ცვლადის- vals სახით.

- **improfile** – გამოითვლის და ააგებს ინტენსივობათა განაწილებას გამონასახზე მითითებული მონაკვეთის გასწვრივ. მონაკვეთი შეგვიძლია განვსაზღვროთ მაუსის საშუალებით, ან მივუთითოთ კოორდინატები, როგორც შემავალი არგუმენტები. ფერადი გამონასახის შემთხვევაში ფუნქცია აგებს ინტენსივობების მნიშვნელობებს შესაბამის ფერებში.

შემდეგი ბრძანებები MATLAB-ში საშუალებას მოგვცემს გამონასახი გამოვიტანოთ გრაფიკულ ფანჯარაში და მოვნიშნოთ მონაკვეთი, რომლის გასწვრივაც გვაინტერესებს პროფილის აგება:

```
imshow peppers.png
improfile
```

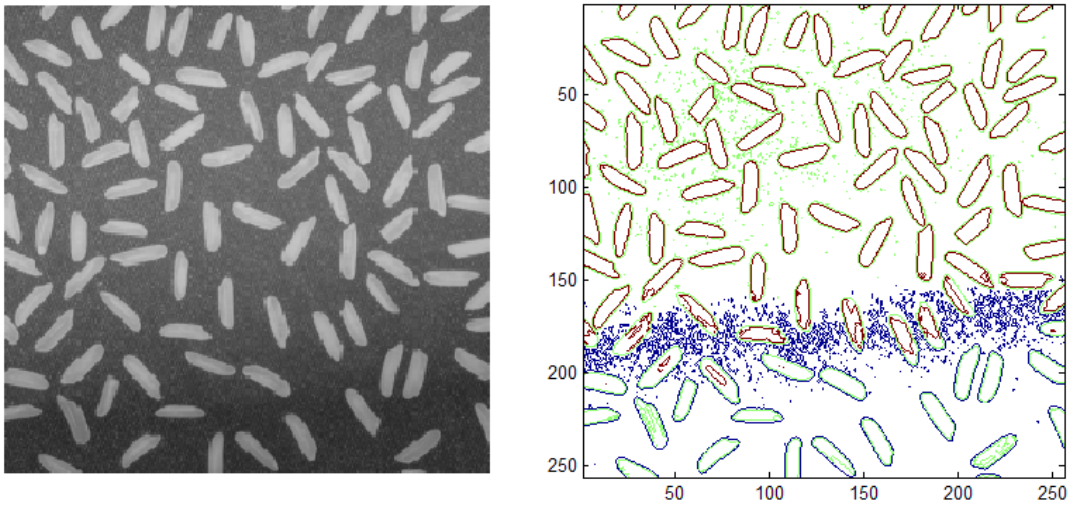


ნახ. 15.10 improfile

- გამონასახის კონტურული გრაფიკი. ფუნქცია **imcontour**

```
I = imread('rice.png');
imshow(I)
figure, imcontour(I,3)
```

ბრძანებათა მწკრივი გამოიძახებს, გრაფიკულ ფანჯარაში გამოიტანს გამონასახს და ახალ გრაფიკულ ფანჯარაში მოგვცემს გამონასახის კონტურულ გრაფიკს.

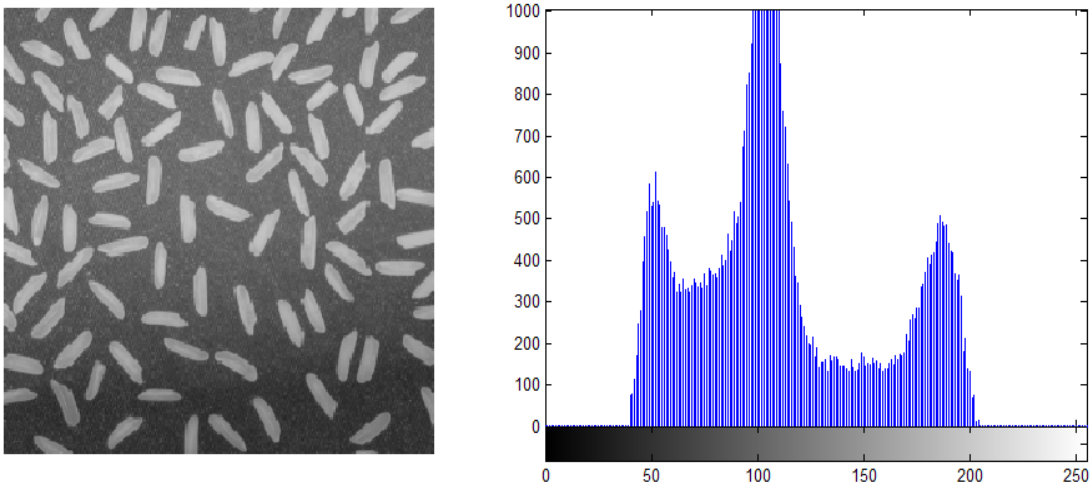


ნახ. 15.11 საწყისი გამონასახი და კონტურული გრაფიკი

- გამონასახის ჰისტოგრამა. ფუნქცია imhist

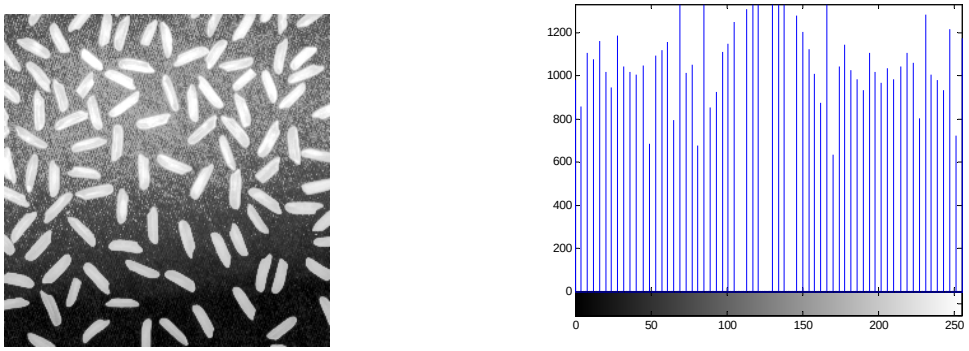
```
I = imread('rice.png');
imshow(I)
figure, imhist(I)
```

ბრძანებათა მწკრივი გამოიძახებს, გრაფიკულ ფანჯარაში გამოიტანს გამონასახს და ახალ გრაფიკულ ფანჯარაში მოგვცემს გამონასახის ჰისტოგრამას.



ნახ. 15.12 საწყისი გამონასახი და ჰისტოგრამა

არსებობს რამდენიმე გზა გამონასახის კონტრასტის შესაცვლელად. ერთ-ერთი მათგანია ფუნქცია **histreq** რაც საშუალებას იძლევა განვავრცოთ ინტენსივობა თანაბრად მთელ გამონასახზე. ამ პროცესს ჰისტოგრამის გათანაბრებას უწოდებენ (histogram equalization).

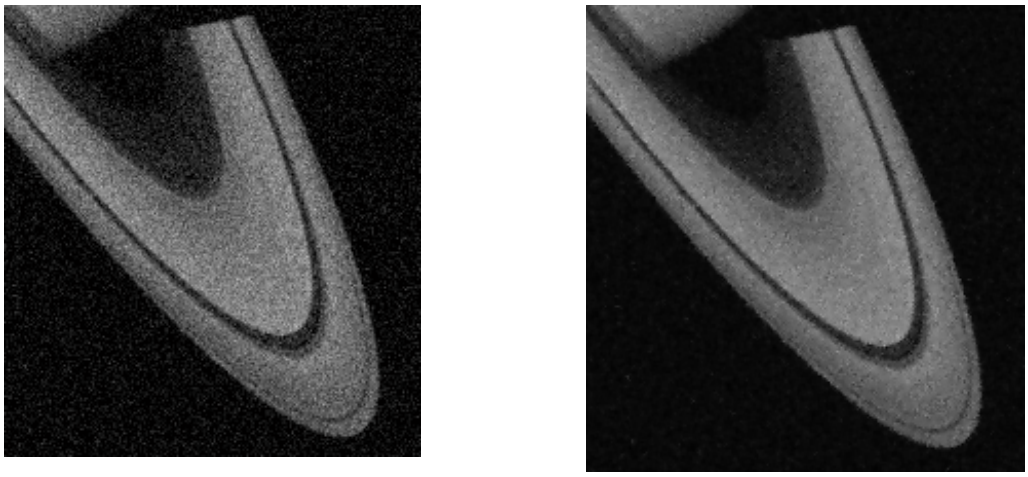


ნახ. 15.13 გათანაბრებული გამონასახი და ჰისტოგრამა

- გამონასახის სტატისტიკური ანალიზი
  ფუნქციები **mean2**, **std2** გამოითვლის გამონასახის შემქმნელი მატრიცის ელემენტების მნიშვნელობათა საშუალოსა და სტანდარტულ გადახრას, ხოლო corr2 ერთნაირი ზომის ორი მატრიცის კორელაციის კოეფიციენტს გამოითვლის.

- ციფრული გამონასახი შესაძლოა დამახინჯებული იყო სხვადასხვა მიზეზით გამოწვეული ხმაურით. MATLAB გააჩნია ხმაურის გამორიცხვის სხვადასვა საშუალება, როგორიცაა წრფივი, მედიანური და ადაპტური ფილტრაცია. ფუნქცია **wiener2** აანალიზებს გამონასახს და ხმაურის ფილტრაციას იმის მიხედვით შეასრულებს. ბრძანებათა შემდეგი მწკრივი გამოიძახებს გამონასახს, გადაიყვანს მას შავ-თეთრ გამონასახში, დაუმატებს ხმაურს. (გაუსის განაწილება) და შემდეგ გაასუფთავებს გამონასახს ხმაურისაგან **wiener2** ფუნქციის საშუალებით.

```
RGB = imread('saturn.png');
I = rgb2gray(RGB);
J = imnoise(I,'gaussian',0,0.005);
K = wiener2(J,[5 5]);
imshow(J)
figure, imshow(K)
```

- 

ნახ. 15.14 გამონასახის გასუფთავება ხმაურისაგან

- გამონასახის ჩაწერა დისკზე

ფუნქცია **imwrite** ჩაწერს გამონასახს მითითებულ ადგილზე.

```
imwrite(I2, 'rice2.png');
```

ხოლო ფუნქცია **iminfo** გვიჩვენებს ინფორმაციას ახლადშექმნილი გამონასახის შესახებ:

```
imfinfo('rice2.png ')

ans =

                    Filename: 'rice2.png '
                 FileModDate: '08-Nov-2010 18:04:56'
                    FileSize: 49552
                      Format: 'png'
               FormatVersion: []
                       Width: 256
                      Height: 256
                    BitDepth: 8
                   ColorType: 'grayscale'
             FormatSignature: [137 80 78 71 13 10 26 10]
                    olormap: []
                   Histogram: []
              InterlaceType: 'none'
                Transparency: 'none'
   SimpleTransparencyData: []
```

```
BackgroundColor: []
RenderingIntent: []
 Chromaticities: []
          Gamma: []
    XResolution: []
    YResolution: []
 ResolutionUnit: []
        XOffset: []
        YOffset: []
     OffsetUnit: []
SignificantBits: []
   ImageModTime: '8 Nov 2010 14:04:56 +0000'
          Title: []
         Author: []
    Description: []
      Copyright: []
   CreationTime: []
       Software: []
     Disclaimer: []
        Warning: []
         Source: []
        Comment: []
      OtherText: []
```

15.9 MATLAB ბრძანებები და ფუნქციები

A=imread(‘filename’)	გამოიძახებს ორფეროვანი(შავ-თეთრი) გამონასახის შემცველ ფაილს და წარმოადგენს მას A მატრიცის სახით (ინდექსირებული გამონასახი)
[X,map]=imread(‘filename’)	გამოიძახებს ფერად გამონასახის შემცველ ფაილს და წარმოადგენს მას A მატრიცის სახით, რომლის შესაბამის ინფორმაციას ფერთა შესახებ წარმოადგენს map მატრიცის სახით
imshow(‘filename’)	წარმოადგენს გამონასახის შემცველ ფაილს გრაფიკულ ფანჯარაში
B=imresize(A,m)	გვაძლევს A ინდექსირებული გამონასახის m-ჯერ გადიდებულ გამონასახს (თუ 0<m<1, გამონასახი შემცირდება)
imrotate(A,angle)	აბრუნებს A გამონასახს ‘angle’ გრადუსით საათის ისრის საწინააღმდეგო მიმართულებით
imadd	გამონასახების შეკრება
imdevide	გამონასახების გაყოფა
imadd	გამონასახების შეკრება
imdevide	გამონასახების გაყოფა
imsubtract	გამონასახების გამოკლება
immultiply	გამონასახების გამრავლება
imabsdiff	გვაძლევს ახალ მატრიცას, რომლის ელემენტებიც არგუმენტთა შესაბამისი ელემენტების სხვაობათა

	აბსოლუტურ სიდიდეს წარმოადგენს
imcrop	ამოჭრის დეტალს გამონასახიდან
impixel	იძლევა ინფორმაციას მონიშნული პიქსელის შესახებ
imcontour	გამონასახის კონტურუკი გრაფიკი
imhist	აგებს გამონასახის ჰისტოგრამას
mean2	mean2 გამოითვლის გამონასახის შემქმნელი მატრიცის ელემენტების მნიშვნელობათა საშუალოს
std2	std2 გამოითვლის გამონასახის შემქმნელი მატრიცის ელემენტების მნიშვნელობათა სტანდარტულ გადახრას,
corr2	გამოითვლის ერთნაირი ზომის ორი მატრიცის კორელაციის კოეფიციენტს.
wiener	გამონასახის ხმაურისაგან გასუფთავება
improfile	აგებს ინტენსივობათა პროფილეს გამონასახზე მითითებილი მონაკვეთის გასწვრივ
dither	გარდაქმნის შავ-თეთრი გამონასახს ბინარულში, ან რეალურს - ინდექსირებულში
gray2ind	გარდაქმნის შავ-თეთრ გამონასახს ინდექსირებულში
im2bw	გარდაქმნის შავ-თეთრ, ინდექსირებულ ან რეალურ გამონასახს ორობით გამონასახში
ind2gray	გარდაქმნის ინდექსირებულ გამონასახს შავ-თეთრში
ind2grb	გარდაქმნის ინდექსირებულ გამონასახს რეალურ გამონასასში
mat2gray	მონაცემთა მატრიცას წარმოადგენს შავ-თეთრ გამონაახის სახით მონაცენთა სკალირების საფუძველზე
rgb2grai	გარდაქმნის რეალურ გამონასახს შავ-თეთრში
rgb2ind	გარდაქმნის რეალურ გამონასახს ინდექსირებულში
imwrite	ჩაწერს გამონასახს მითითებულ ადგილზე
iminfo	გვიჩვენებს ინფორმაციას გამონასახის შესახებ