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
```

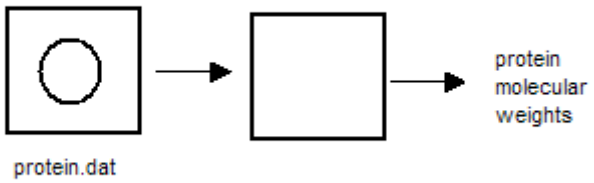
შეესაბამება ცილას ამინომჟავათა მიმდევრობით – LysGluMetAspSerGlu.

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:

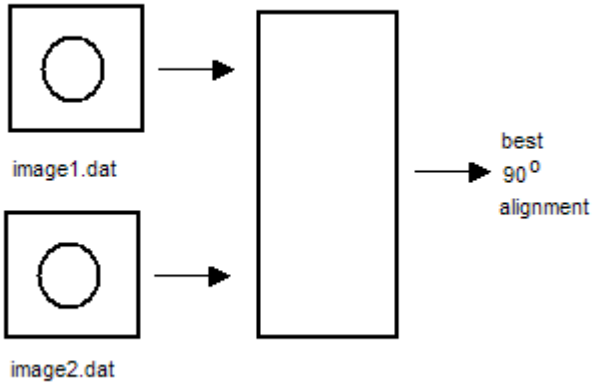
```
distance = sum(sum(image1 – image2).^2));
```

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.