Міністерство освіти і науки України
Вінницький національний технічний університет

В. М. Михалевич

Maple.
Комп'ютерна підтримка курсу вищої математики в технічному вузі

Частина I
Лінійна й векторна алгебра. Аналітична геометрія

Затверджено Вченою радою Вінницького національного технічного університету як навчальний посібник для студентів усіх спеціальностей. Протокол N 7 від " 4 " 03 2004 р.

Вінниця ВНТУ 2004

УДК 681.3.06+514.122
М 76

Рецензенти:

В. О. Капустян, доктор фізико-математичних наук професор
В.І. Клочко, доктор педагогічних наук, професор
С. В. Юхимчук, доктор технічних наук, професор

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України

Михалевич В. М.

М 76 **Maple. Комп'ютерна підтримка курсу вищої математики в технічному вузі. Частина І. Лінійна й векторна алгебра. Аналітична геометрія.** Навчальний посібник. - Вінниця: ВНТУ, 2004. - 111 с.

Навчальний посібник присвячений знайомству з можливостями пакета Maple при розв’язуванні задач лінійної й векторної алгебра та аналітичної геометрії. Приведено програми, які розроблені в середовищі пакета Maple та дозволяють не тільки отримати кінцевий результат, а й зрозуміти хід розв’язання деяких типових задач. Матеріал подається у послідовності, прийнятій для багатьох робочих навчальних программ з вищої математики, розроблених у ВНТУ. Приведено приклади електронних лекцій, створенних в середовищі пакета Maple.

Розрахований на студентів технічних ВНЗ усіх форм навчання та спеціальностей.

УДК 681.3.06+514.122

ВВЕДЕННЯ

Можливості сучасних математичних пакетів настільки вражаючі, що традиційні методики викладання з “крейдою та ганчіркою” уже не можуть вважатися сучасними. Необхідність впровадження комп’ютерних технологій навчання, зокрема технологій, що базуються на використанні математичних пакетів, ні у кого не викликає сумніву. Основне питання полягає в тому, як це зробити з якомога більшим ефектом в реальних умовах. Очевидно, що перш за все потрібно, по можливості найкраще, вивчити ці пакети. Справа в тому, що створювалися найбільш популярні сучасні пакети в першу чергу для професійної наукової або інженерної діяльності. Тому використання подібних пакетів в освіті являє собою “непахане поле”. Уяву про сучасний стан питання в цьому напрямі можна отримати, зазирнувши на сайт http://www.exponenta.ru.

Цей посібник присвячений знайомству з можливостями пакета Maple.

У найбільш популярних книжках [1-7] акцент, як правило, ставиться на описанні команд Maple з прикладами розв’язання математичних задач. У цьому посібнику описання можливостей Maple виконується з точки зору програми вищої математики у технічному вузі. Спочатку автор хотів написати цикл лекцій у середовищі Maple так, щоб лекції з вищої математики можна було б проводити в комп’ютерному залі: кожен студент сидить за комп’ютером, завантажує електронний варіант лекції, прослуховує стислий оглядовий вступ лектора, а потім самостійно працює з “живими сторінками” (термін “живі сторінки” буде пояснено далі), маючи змогу отримати миттєву консультацію викладача. За таких умов стає розпливчатою межа між традиційними формами проведення занять: лекції, практичні заняття, лабораторні роботи. Але для того, щоб писати подібні електронні підручники, необхідне досить широке та глибоке знайомство з Maple. Причому на перший план часто виступають не проблеми розв’язання тієї або іншої задачі, а подання матеріалу у традиційній формі представлення проміжних результатів – візуалізація деяких даних. Наприклад, за допомогою Maple дуже легко можна отримати розв’язок системи лінійних алгебраїчних рівнянь. Але задаються рівняння системи у вигляді послідовності цих рівнянь, записаних у рядок:

```
> ISys:={-3*a+2*b-c=2,-a+b+c=0,2*a+b+2*c=2};
```

І так само виводяться на екран дисплея

$$Sys := \{-3\,a + 2\,b - c = 2, -a + b + c = 0, 2\,a + b + 2\,c = 2\}$$

Отримати запис рівнянь у формі, близькій до традиційної, можна за допомогою організації циклу

```
> for i from 1 to nops(Sys) do
    Sys[i]
end do;
```

Результатом виконання даного циклу буде виведення на екран дисплея рівнянь системи у такій формі:

$$-3\,a + 2\,b - c = 2$$

$$-a + b + c = 0$$

$$2\,a + b + 2\,c = 2$$

Для того ж, щоб отримати зліва фігурну дужку, потрібна ще більша винахідливість, оскільки це можна зробити тільки нестандартним застосуванням команди задання кусково-неперервних функцій:

```
> piecewise(Sys[1],``,Sys[2],``,Sys[3],``);
```

$$\begin{cases} -3\,a + 2\,b - c = 2 \\ -a + b + c = 0 \\ 2\,a + b + 2\,c = 2 \end{cases}$$

Безумовно, подібна ситуація створює певні незручності. Але маємо те, що маємо. А маємо унікальний пакет аналітичних обчислень *Maple*, який має власну мову програмування, що дозволяє, при певному досвіді, досить легко створювати суперкласні навчальні програми.

Отже, перед тим як писати електронні лекції у середовищі Maple, потрібно розробити відповідні інструменти - цеглинки, за допомогою яких ця робота може бути виконана досить ефективно. Саме розробці такого інструментарію і присвячений цей посібник.

Значна увага приділена розробці програм, за допомогою яких отримується не просто відповідь, а **хід розв'язання деяких типових задач**. Причому матеріал подається у послідовності, прийнятій для багатьох робочих навчальних программ з вищої математики, розроблених у ВНТУ. Як правило, дається стисле пояснення команд Maple, які застосовуються при розв'язанні конкретних задач. Але ця книга не тільки ні в якій мірі не претендує на опис програмного продукту Maple, а навпаки, автор намагався уникнути формального описання команд Maple, яке можна знайти в довідковій системі, або в інших спеціально написаних для цього книжках [1-7]. Цей посібник не може замінити для студентів і навчальну літературу з вищої математики, але може суттєво допомогти більш кращому та швидшому засвоєні цього курсу.

Цей посібник підготовлено на основі електронного варіанта посібника, написаного в середовищі пакета Maple 9. Але всі команди та

програми, що приведені у посібнику, повинні працювати і в пакетах Maple версій 6, 7, 8. Більш того, значна частина наведеного програмного коду буде працювати і в Maple V R4 і, навіть, в DEMO версії Maple. Причому, переважна частина Maple-коду при відповідній модифікації буде цілком придатна для роботи в DEMO версії. Це особливо важливо, оскільки пакет Maple досить дорогий, навіть з урахуванням знижок для університетів. Інсталяційний пакет DEMO-Maple займає 1452 Кб і може бути встановлений на всіх комп'ютерах університету та на приватних комп'ютерах студентів. Автором посібника розроблено технологію підготовки студентами файлів для роботи в DEMO версії та збереження результатів роботи, яка дозволяє частково "обійти" обмеження DEMO версії, в середовищі якої заблоковано операції копіювання, вставки та збереження файлу. Значно більш серйозні обмеження DEMO версії містяться в кількості доступних команд і функцій. Деякі з них також вдається «обійти». Саме за допомогою програм, приклади яких наведено в посібнику.

Значна частина посібника, зокрема більшість приведених програм, є авторською розробкою.

1. Матричні і векторні обчислення

Середовище Maple дозволяє виконувати всі стандартні операції, визначені в лінійній алгебрі. Вони стають доступні при підключенні бібліотеки linalg або більш нового пакета LinearAlgebra.

1.1 Створення матриць і векторів

Для задання матриці використовується команда matrix. Причому задавати матрицю можна різними способами, як видно з декількох прикладів:

```
restart:
with(linalg):
A:=matrix([[1,1,1],[4,1,6],[7,1,9]]);#Квадратна матриця 3 порядку
a:=matrix([[2,x^2,4,5.3,alpha]]);#Матриця - рядок
A1:=matrix(4,4,[seq(k^2,k=3..18)]);#Квадратна матриця 4 порядку
a1:=matrix(3,3,-10);
```

$$A := \begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \end{bmatrix}$$

$$a := [2 \quad x^2 \quad 4 \quad 5.3 \quad \alpha]$$

$$A1 := \begin{bmatrix} 9 & 16 & 25 & 36 \\ 49 & 64 & 81 & 100 \\ 121 & 144 & 169 & 196 \\ 225 & 256 & 289 & 324 \end{bmatrix}$$

$$a1 := \begin{bmatrix} -10 & -10 & -10 \\ -10 & -10 & -10 \\ -10 & -10 & -10 \end{bmatrix}$$

Відомо, що в Maple (як і в багатьох інших програмних продуктах) під матрицею розуміється двовимірний масив, індекс якого змінюється від одиниці до будь-якого цілого числа. Отже, матрицю також можна задати таким чином:

```
> AA:=array(1..2,1..3);
```

$$AA := \text{array}(1 .. 2, 1 .. 3, [\ ])$$

Але масив вигляду **array(1..2,0..3)** матрицею не є, оскільки другий індекс змінюється від нуля.

Є чудові можливості для формування матриць спеціального вигляду. Наприклад, сформувати одиничну матрицю можна таким чином:

> **E:=array(identity,1..2,1..2): `E`=evalm(E);**

$$E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Для отримання діагональної матриці можна використовувати команду diag(seq), де seq - послідовність елементів, розташованих на головній діагоналі. Відмітимо, якщо seq складається з одиниць, то отримана матриця буде одиничною.

> **De:=diag(1,2,3);**

$$De := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}.$$

Скалярну матрицю можна отримати так:

> **Sk1:=diag(seq(lambda,i=1..4));**

$$Sk1 := \begin{bmatrix} \lambda & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 0 & 0 & \lambda & 0 \\ 0 & 0 & 0 & \lambda \end{bmatrix}$$

або як добуток деякого числа λ на одиничну матрицю E, яка була створена вище

> **Sk:=lambda*E:Sk2:=evalm(Sk);**

$$Sk2 := \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

В першому випадку ми скористалися оператором **seq** для генерації послідовності значень. Більш докладну інформацію про цей оператор з прикладами його використовування можна знайти в довідковій системі Maple. Для отримання довідки про конкретний оператор необхідно клацнути курсором мишки на ньому і натиснути клавішу F1.

Будь-який елемент матриці можна отримати або перевизначити, вказавши номери рядка і стовпця, на перетині яких знаходиться даний елемент:

> **B:=matrix([[1,1,1],[4,1,6],[7,1,9],[-2,1,0]]);**
b[2,1]:=B[2,1];b[4,3]:=B[4,3];B[4,3]:=-5;`B`=evalm(B);

$$B := \begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \\ -2 & 1 & 0 \end{bmatrix}$$

$$b_{2,1} := 4$$

$$b_{4,3} := 0$$

$$B_{4,3} := -5$$

$$B = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \\ -2 & 1 & -5 \end{bmatrix}$$

Для показу у вікні дисплея всіх елементів матриці використано оператор evalm. Якщо спробувати отримати неіснуючий елемент матриці, буде отримано повідомлення:

```
> B[2,4];
Error, 2nd index, 4, larger than upper array bound 3
```

"Помилка, 2-ий індекс дорівнює 4, що перевищує число стовпців матриці, рівне 3."

1.2 Дії над матрицями.

```
> restart:
with(linalg):
```

1.2.1 Множення матриці на число або числа на матрицю:

```
> A:=matrix([[1,1,1],[4,1,6],[7,1,9],[-2,1,0]]);
gamma*A=evalm(gamma*A);`3*A`=evalm(3*A);`A*3`=evalm(A*
3);
```

$$A := \begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \\ -2 & 1 & 0 \end{bmatrix}$$

$$\gamma A = \begin{bmatrix} \gamma & \gamma & \gamma \\ 4\gamma & \gamma & 6\gamma \\ 7\gamma & \gamma & 9\gamma \\ -2\gamma & \gamma & 0 \end{bmatrix}$$

$$3*A = \begin{bmatrix} 3 & 3 & 3 \\ 12 & 3 & 18 \\ 21 & 3 & 27 \\ -6 & 3 & 0 \end{bmatrix}$$

$$A*3=\begin{bmatrix} 3 & 3 & 3 \\ 12 & 3 & 18 \\ 21 & 3 & 27 \\ -6 & 3 & 0 \end{bmatrix}$$

1.2.2 Складання матриць

```
> A:=matrix([[1,1,1],[4,1,6]]);
B:=matrix([[-1,-1,-1],[4,0,-2]]);
C:=evalm(A&+B):`C=A+B`=evalm(C);
`B-A`=evalm(B-A);
```

$$A:=\begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \end{bmatrix}$$

$$B:=\begin{bmatrix} -1 & -1 & -1 \\ 4 & 0 & -2 \end{bmatrix}$$

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

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

1.2.3 Транспонування матриці

Транспонування матриці A виконується за допомогою команди transpose(A):

```
> `transpose(C)`=transpose(C);
```

$$transpose(C)=\begin{bmatrix} 0 & 8 \\ 0 & 1 \\ 0 & 4 \end{bmatrix}$$

1.2.4 Множення матриць

Операція *множення матриць* дещо складніша.

Для розуміння суті цієї операції розглянемо приклад множення матриці-рядка на матрицю-стовпець:

```
> dimColRow:=4:m:=1:r:=1:
a:=matrix(m,dimColRow):b:=matrix(dimColRow,r):
`a`=evalm(a);`b`=evalm(b);
print(`c=a*b`);c:=evalm(a&*b);
```

$$a=[a_{1,1} \quad a_{1,2} \quad a_{1,3} \quad a_{1,4}]$$

$$b=\begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \\ b_{4,1} \end{bmatrix}$$

$$c=a*b$$

$$c := [a_{1,1}\,b_{1,1} + a_{1,2}\,b_{2,1} + a_{1,3}\,b_{3,1} + a_{1,4}\,b_{4,1}]$$

або

```
> dimColRow:=3:m:=3:r:=2:
a:=matrix(m,dimColRow):b:=matrix(dimColRow,r):
`a`=evalm(a);`b`=evalm(b);
print(`c=a*b`);c:=evalm(a&*b);
```

$$a = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

$$b = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix}$$

$$c=a*b$$

$$c := \begin{bmatrix} a_{1,1}\,b_{1,1} + a_{1,2}\,b_{2,1} + a_{1,3}\,b_{3,1} & a_{1,1}\,b_{1,2} + a_{1,2}\,b_{2,2} + a_{1,3}\,b_{3,2} \\ a_{2,1}\,b_{1,1} + a_{2,2}\,b_{2,1} + a_{2,3}\,b_{3,1} & a_{2,1}\,b_{1,2} + a_{2,2}\,b_{2,2} + a_{2,3}\,b_{3,2} \\ a_{3,1}\,b_{1,1} + a_{3,2}\,b_{2,1} + a_{3,3}\,b_{3,1} & a_{3,1}\,b_{1,2} + a_{3,2}\,b_{2,2} + a_{3,3}\,b_{3,2} \end{bmatrix}$$

Матриці співмножники повинні задовольняти умову: кількість стовпців першої матриці повинна дорівнювати кількості рядків другої матриці. В результаті дістаємо матрицю розмірності (*m* x *n*), де m - кількість рядків першої матриці, *n* - кількість стовпців другої матриці:

```
> A:=matrix(2,3,[1,2,-3,3,2,0]):
B:=matrix(3,2,[3,1,2,4,-1,1]):
A=evalm(A);B=evalm(B);`A*B`=evalm(A&*B);
```

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

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

$$A*B = \begin{bmatrix} 10 & 6 \\ 13 & 11 \end{bmatrix}$$

Якщо ця умова не виконується, буде отримано повідомлення

```
> `b*c`=evalm(b&*c);
Error, (in linalg[multiply]) non matching dimensions for
vector/matrix product
```

"помилка, невідповідність розмірності для операції множення матриць".

Щоб взнати кількість рядків або стовпців матриці, необхідно

виконати такі операції:

```
> `количество строк матрицы A`=rowdim(A);
`количество столбцов матрицы A`=coldim(A);
> print(`Размерность матрицы A`);
matrix(1,3,[rowdim(A),`*`,coldim(A)]);
```

количество строк матрицы Sk1=2

количество столбцов матрицы A=3

Размерность матрицы A

[2 * 3]

1.3 Визначники

1.3.1 Обчислення визначника

```
> restart:
with(linalg):
Warning, the protected names norm and trace have been
redefined and unprotected
```

Для обчислення визначника використовується команда **det**

```
> A:=matrix(2,3,[1,2,-3,3,2,0]):
B:=matrix(3,2,[3,1,2,4,-1,1]):
`det(A*B)`=det(A&*B);
```

$$det(A*B) = 32$$

Формулу обчислення визначника в символьному вигляді можна отримати таким чином:

```
> m:=2:a:=matrix(m,m):`a`=evalm(a);
```

$$a = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}$$

```
> `det(a)`=det(a);
```

$$det(a) = a_{1,1}\,a_{2,2} - a_{1,2}\,a_{2,1}$$

Змінивши значення **m** на **3** або більше ціле число, отримаємо формулу для обчислення визначника більш високого порядку.

Для оволодіння технікою обчислення визначника за формулою розкладання можна скористатися командами визначення мінора minor(A,n,m). Однак тут треба бути уважним. В Maple під мінором

розуміється матриця, отримана вилученням n - го рядка і m - го стовпця вихідної матриці.

```
> m:=3:a:=matrix(m,m):
`a`=evalm(a);`minor(a,2,3)`=minor(a,2,3);
```

$$a = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

$$minor(a,2,3) = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{3,1} & a_{3,2} \end{bmatrix}$$

Для отримання істинного мінора, тобто визначника відповідної матриці, необхідно доповнити стандартну команду

```
> det(minor(a,2,3));
```

$$a_{1,1}\,a_{3,2} - a_{1,2}\,a_{3,1}$$

Алгебраїчне доповнення нескладно визначити так

```
> A[2,3]:=(-1)^(2+3)*det(minor(a,2,3));
```

$$A_{2,3} := -a_{1,1}\,a_{3,2} + a_{1,2}\,a_{3,1}$$

Освоєння техніки обчислення визначника за формулою розкладання нерідко викликають труднощі для студентів. Причому арифметичні помилки перемішуються із змістовними. Тому студенту самому часто буває важко знайти свої помилки. В результаті студенти змушені невиправдано багато часу приділяти для освоєння відповідної техніки. І справу тут навіть не в тому, що викладач не завжди поряд, а головне, що подібні питання, яких з курсу вищої математики більше ніж достатньо, змушують викладача працювати в режимі рутинних перевірок, тим самим знижуючи ефективність його роботи. Якщо ж дати студентові інструмент, за допомогою якого він сам зможе себе перевіряти, то в результаті не тільки викладач буде звільнений від частини рутинної роботи, а й робота самого студента буде більш ефективною за рахунок її більшої активності та самостійності.

Для отримання символьного виду виразу для обчислення визначника за формулою розкладання може бути використана спеціально розроблена автором процедура **det_matrixS**, яка приведена нижче. Варто звернути увагу, що основна формула розкладання формується в циклі одним із двох однотипних операторів:

```
y:=sum('(-1)^(j+i)*a[j,i]*det*
```

```
        linalg['minor'](a,j,i)','i'=1..m):
```

або

```
    y:=sum('(-1)^(j+i)*a[i,j]*det*
        linalg['minor'](a,i,j)','i'=1..m):
```

Переважна більшість решти операторів призначена для стандартних перевірок правильності задання аргументів процедури.

```
> det_matrixS:=proc(a::matrix,j::integer,
RowOrCol::string)
local y,i,m,n,detM,c;
m:=`linalg/rowdim`(a);
n:=`linalg/coldim`(a);
if m<>n then
     error("Матриця %1 повинна бути квадратною",a);
end if;
if (j<1) then
     error "Число j повинне бути натуральним числом,
що не перевищує m"
end if;
if m=1 then
  return(a[1,1]);
end if;
if RowOrCol="R" then
   y:=sum('(-1)^(j+i)*a[j,i]*det*
linalg['minor'](a,j,i)',
'i'=1..m):
else
   y:=sum('(-1)^(j+i)*
a[i,j]*det*linalg['minor'](a,i,j)',
'i'=1..m):
end if
end proc;
```

```
b:=matrix(3,3,[1,2,3,4,5,6,7,8,-9]):print(B=evalm(b));
d1:=matrix(4,4,[1,2,3,4,5,6,7,8,-9,1,-
4,5,0,7,2,9]):print(D1=evalm(d1));
```

При виклику процедури необхідно вказати матрицю, номер ряду, по якому буде проведено розкладання, і вказати вид ряду, - горизонтальний ("R"-"Row") або вертикальний ("C" - "Column")

```
> `det(B)`=det_matrixS(b,2,"R");
`det(B)`=det_matrixS(b,3,"C");
`det(D1)`=det_matrixS(d1,4,"C");
```

$$det(B) = -4\ det\begin{bmatrix}2 & 3\\ 8 & -9\end{bmatrix} + 5\ det\begin{bmatrix}1 & 3\\ 7 & -9\end{bmatrix} - 6\ det\begin{bmatrix}1 & 2\\ 7 & 8\end{bmatrix}$$

$$det(B) = 3\ det\begin{bmatrix}4 & 5\\ 7 & 8\end{bmatrix} - 6\ det\begin{bmatrix}1 & 2\\ 7 & 8\end{bmatrix} - 9\ det\begin{bmatrix}1 & 2\\ 4 & 5\end{bmatrix}$$

$$det(D1) = -4\ det\begin{bmatrix}5 & 6 & 7\\ -9 & 1 & -4\\ 0 & 7 & 2\end{bmatrix} + 8\ det\begin{bmatrix}1 & 2 & 3\\ -9 & 1 & -4\\ 0 & 7 & 2\end{bmatrix} - 5\ det\begin{bmatrix}1 & 2 & 3\\ 5 & 6 & 7\\ 0 & 7 & 2\end{bmatrix} + 9\ det\begin{bmatrix}1 & 2 & 3\\ 5 & 6 & 7\\ -9 & 1 & -4\end{bmatrix}$$

Розроблена процедура значною мірою ілюструє те, що автор розуміє під терміном “живі сторінки”. Змінивши вихідні дані, в даному випадку елементи матриці, ми автоматично отримаємо і відповідні зміни у всьому ході знаходження визначника. На основі приведеної процедури нескладно створити іншу процедуру, яка дає, так би мовити, не тільки перший рівень формули розкладання, а й додатково формули для обчислення всіх визначників до другого порядку включно, наприклад

$$det(B) = -4\ det\begin{bmatrix}2 & 3\\ 8 & -9\end{bmatrix} + 5\ det\begin{bmatrix}1 & 3\\ 7 & -9\end{bmatrix} - 6\ det\begin{bmatrix}1 & 2\\ 7 & 8\end{bmatrix}$$

$$det\begin{bmatrix}2 & 3\\ 8 & -9\end{bmatrix} = 2\ det\,[\,-9] - 3\ det\,[\ 8]$$

$$det\begin{bmatrix}1 & 3\\ 7 & -9\end{bmatrix} = det\,[\,-9] - 3\ det\,[\ 7]$$

$$det\begin{bmatrix}1 & 2\\ 7 & 8\end{bmatrix} = det\,[\ 8] - 2\ det\,[\ 7]$$

За допомогою приведеної процедури можна отримати формулу для обчислення визначника в символьному вигляді

```
> a:=matrix(3,3):print(A=evalm(a));
`det(A)`=det_matrixS(a,1,"R");
```

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

$$det(A) = a_{1,1}\, det\begin{bmatrix} a_{2,2} & a_{2,3} \\ a_{3,2} & a_{3,3} \end{bmatrix} - a_{1,2}\, det\begin{bmatrix} a_{2,1} & a_{2,3} \\ a_{3,1} & a_{3,3} \end{bmatrix} + a_{1,3}\, det\begin{bmatrix} a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{bmatrix}$$

або

```
> a:=matrix(4,4):print(A=evalm(a));
`det(A)`=det_matrixS(a,1,"C");
```

$$A=\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix}$$

$$det(A) = a_{1,1}\, det\begin{bmatrix} a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} - a_{2,1}\, det\begin{bmatrix} a_{1,2} & a_{1,3} & a_{1,4} \\ a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix}$$
$$+ a_{3,1}\, det\begin{bmatrix} a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,2} & a_{2,3} & a_{2,4} \\ a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} - a_{4,1}\, det\begin{bmatrix} a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,2} & a_{3,3} & a_{3,4} \end{bmatrix}$$

Для обчислення визначника можна скласти і власну процедуру

```
> det_matrix:=proc(a::matrix)
local i,m,n,detM,c;
m:=`linalg/rowdim`(a);
n:=`linalg/coldim`(a);
if m<>n then
     error("Матриця %1 повинна бути квадратною",a);
end if;
if m=1 then
  return(a[1,1]);
end if;
detM:=0;
for i from 1 to m do
c:=linalg['minor'](a,i,1);
#c:=linalg['delrows'](linalg['submatrix'](a,
[seq(k,k=1..m)], [seq(k,k=2..m)]),i..i);
#c:=`linalg/delcols`(`linalg/delrows`(a,i..i),1..1);
```

```
  detM:=detM+(-1)^(1+i)*a[i,1]*det_matrix(c);
end do;
end proc:
```

Обчислимо за допомогою складеної процедури визначник матриці **A**

```
> A:=matrix(3,3,[[2,1,0],[-3,4,0],[-2,1,2]]);
detA=det_matrix(A);
```

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

$$detA = 22$$

В основу створення процедури **det_matrix** покладено ідею рекурсивного звернення до неї при обчисленні мінора. В тексті програми є закоментовані оператори, що відображують можливі альтернативні варіанти визначення матриці **c**. (Всі символи на лінії, які слідують за символом **#,** ігноруються програмою). Приведені варіанти відрізняються і за командами, що використовуються, і за синтаксисом.

1.3.2 Властивості визначника

Для перевірки властивостей визначника підійдуть такі команди. Команди для перестановки місцями двох рядків або двох стовпців - **swaprow**(A,i1,i2) або **swapcol**(A,j1,j2):

```
> A:=matrix([[1,1,1],[4,1,6],[7,1,9]]);
`swaprow(A,1,3)`=swaprow(A,1,3);#Міняються місцями перший і третій рядки
`swapcol(A,2,1)`=swapcol(A,2,1);#Міняються місцями перший і другий стовпці
```

$$A := \begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \end{bmatrix}$$

$$swaprow(A,1,3) = \begin{bmatrix} 7 & 1 & 9 \\ 4 & 1 & 6 \\ 1 & 1 & 1 \end{bmatrix}$$

$$swapcol(A,2,1) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 6 \\ 1 & 7 & 9 \end{bmatrix}$$

При перестановці двох стовпців знак визначника змінюється на протилежний

```
>`det(A)`=det(A);`det(swapcol(A,2,1))`=det(swapcol(A,2
,1));
```

$$det(A) = 6$$

$$det(swapcol(A,2,1)) = -6$$

Природно, що при подвійній перестановці стовпців знак визначника не змінюється

```
>`det(swapcol(swapcol(A,3,1),2,1))`=det(swapcol(swapcol
(A,3,1),2,1));
```

$$det(swapcol(swapcol(A,3,1),2,1)) = 6$$

Помножити стовпець з номером *j* на значення виразу *expr* можна командою mulcol(A,j,expr), а та ж сама дія для рядка з номером *i* виконується командою mulrow(A,i,expr):

```
> `A`=evalm(A);`mulcol(A,1,3)`=mulcol(A,1,3);
```

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \end{bmatrix}$$

$$mulcol(A,1,3) = \begin{bmatrix} 3 & 1 & 1 \\ 12 & 1 & 6 \\ 21 & 1 & 9 \end{bmatrix}$$

Властивість визначника: *загальний множник елементів стовпця можна винести за знак визначника*, добре ілюструє такий рядок

```
➢ det*(mulcol(A,1,beta))-
  beta*det*[evalm(A)]=det(mulcol(A,1,beta))-
  beta*det(A);
```

$$det\begin{bmatrix} \beta & 1 & 1 \\ 4\beta & 1 & 6 \\ 7\beta & 1 & 9 \end{bmatrix} - \beta\, det\begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \end{bmatrix} = 0$$

Команди addrow(A,i,j,m) і addcol(A,i,j,m) призначені для множення ***i*-го** рядка (стовпця) на скалярний множник **m** і додавання до ***j*-го** рядка (стовпця).

```
> A:=matrix([[1,1,1],[4,1,6],[7,1,9]]);
`addcol(A,1,3,-1)`=addcol(A,1,3,-1);
```

$$A := \begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \end{bmatrix}$$

$$addcol(A,1,3,-1) = \begin{bmatrix} 1 & 1 & 0 \\ 4 & 1 & 2 \\ 7 & 1 & 2 \end{bmatrix}$$

Тобто перший і другий стовпець залишаються без змін, а на місце третього стовпця записується вираз (-1)*col(A,1) + col(A,3). Переконатися в цьому можна за допомогою такої команди

> **col3:=evalm((-1)*col(A,1)+col(A,3));**

$$col3 := [0, 2, 2]$$

Неважко перевірити, що визначник матриці після подібних перетворень не змінився

> **det*evalm(A)-det*(addcol(A,1,3,-1))=det(A)-det(addcol(A,1,3,-1));**

$$det\begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \end{bmatrix} - det\begin{bmatrix} 1 & 1 & 0 \\ 4 & 1 & 2 \\ 7 & 1 & 2 \end{bmatrix} = 0$$

Аналогічним чином можна переконатися, що визначник матриці не змінюється і при її транспонуванні

> **det*evalm(A)-det*(transpose(A))=det(A)-det(transpose(A));**

$$det\begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \end{bmatrix} - det\begin{bmatrix} 1 & 4 & 7 \\ 1 & 1 & 1 \\ 1 & 6 & 9 \end{bmatrix} = 0$$

1.4 Обернена матриця

Для знаходження оберненої матриці призначена команда **inverse**(A)

> **inverse(A);**

$$\begin{bmatrix} \frac{1}{2} & \frac{-4}{3} & \frac{5}{6} \\ 1 & \frac{1}{3} & \frac{-1}{3} \\ \frac{-1}{2} & 1 & \frac{-1}{2} \end{bmatrix}$$

Перевіримо отриманий результат:

```
> `A&*inverse(A)=E`=evalm(A&*inverse(A));
```

$$A\&*inverse(A)=E=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

При перемножуванні матриці на обернену ми отримали те, що повинні були отримати - одиничну матрицю.

Для обчислення приєднаної матриці можна скористатися виразом

```
> `matrix A`=evalm(A);
`adjugate matrix A~`=evalm(inverse(A)*det(A));
```

$$matrix\ A=\begin{bmatrix}1 & 1 & 1\\ 4 & 1 & 6\\ 7 & 1 & 9\end{bmatrix}$$

$$adjugate\ matrix\ A\sim=\begin{bmatrix}3 & -8 & 5\\ 6 & 2 & -2\\ -3 & 6 & -3\end{bmatrix}$$

1.5 Розв'язання систем лінійних рівнянь

```
> restart:
with(linalg):
```

Для розв'язання матричного рівняння **AX=B** використовується команда **linsolve**(A,B), де A - матриця, X, B - матриці або вектори. Система лінійних рівнянь, яка задається матрицями

```
> A:=matrix(3,3,[[-3,2,-1],[-1,1,1],[2,1,2]]);
B:=Vector([2,0,2]);
X:=Vector([seq(x[k],k=1..coldim(A))]);
```

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

$$B:=\begin{bmatrix}2\\ 0\\ 2\end{bmatrix}$$

$$X:=\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}$$

має вигляд

```
> c:=evalm(A&*X):
for i to coldim(A) do
   c[i]=B[i]
end do;
```

$$-3\,x_1 + 2\,x_2 - x_3 = 2$$

$$-x_1 + x_2 + x_3 = 0$$

$$2\,x_1 + x_2 + 2\,x_3 = 2$$

Maple дозволяє "намалювати" систему рівнянь в стандартній формі

```
> eqi:=[seq( c[i]=B[i], i=1..3 )]:
piecewise(eqi[1],``,eqi[2],``,eqi[3],``);
```

$$\begin{cases} -3\,x_1 + 2\,x_2 - x_3 = 2 \\ -x_1 + x_2 + x_3 = 0 \\ 2\,x_1 + x_2 + 2\,x_3 = 2 \end{cases}$$

Розв'язок системи можна отримати використанням команди **linsolve**(A, B)

```
> X:=linsolve(A, B);
```

$$X := [1, 2, -1]$$

1.5.1 Матричний метод розв'язання системи лінійних рівнянь

Матричний метод може бути реалізований за допомогою декількох команд, що вже використовувалися:

```
> invA:=evalm(inverse(A));#Знаходження оберненої матриці для матриці системи
`X=invA*B`=evalm(invA)*`*`*B;#Символічний запис стовпця невідомих у вигляді добутку двох матриць
X:=evalm(invA&*B);#Обчислення вектора невідомих
Подання вектора X у вигляді вектора-стовпця
X:=convert(X,Vector):'X'=LinearAlgebra[Transpose] (X);
```

$$invA := \begin{bmatrix} \frac{1}{8} & \frac{-5}{8} & \frac{3}{8} \\ \frac{1}{2} & \frac{-1}{2} & \frac{1}{2} \\ \frac{-3}{8} & \frac{7}{8} & \frac{-1}{8} \end{bmatrix}$$

$$X=invA*B=\begin{bmatrix} \frac{1}{8} & \frac{-5}{8} & \frac{3}{8} \\ \frac{1}{2} & \frac{-1}{2} & \frac{1}{2} \\ \frac{-3}{8} & \frac{7}{8} & \frac{-1}{8} \end{bmatrix} * \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix}$$

$$X := [\,1, 2, -1\,]$$

$$X=\begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$$

1.5.2 Метод Крамера розв'язання систем лінійних алгебраїчних рівнянь

```
> A:=matrix(3,3,[[-3,2,-1],[-1,1,1],
[2,1,2]]);#Формування матриці коефіцієнтів системи
B:=matrix([[2],[0],[2]]);#Формування матриці-стовпця вільних
членів
AB:=concat(A,B):#Формування розширеної матриці системи
detA:=det(A):print('detA'=Delta,Delta=detA);x:='x':x1:
='x1':
P:=array([[seq(x[k],k=1..coldim(A))]]):
Px:=array([[seq(x[k],k=1..coldim(A))]]):
for i to coldim(A) do
#Формування матриць заміною i-го стовпця матриці А системи стовпцем
вільних коефіцієнтів В
```

Команда **swapcol** міняє місцями *i* - ий стовпець з останнім стовпцем (стовпець *B*), а команда **delcols** вилучає *i* - ий стовпець, що став четвертим

```
    Delta[i]:=delcols(swapcol(AB,4,i),4..4);

print(Delta[i]='det'(delcols(swapcol(AB,4,i),4..4)));

print(Delta[i]=det(Delta[i]),Px[1,i]=det(Delta[i])/det
A):#Обчислення невідомих за формулами Крамера
P[1,i]:=det(Delta[i])/detA;
end do:
transpose(Px)=transpose(evalm(P));
```

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

$$B := \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix}$$

$$detA = \Delta,\ \Delta = 8$$

$$\Delta_1 = \det\left(\begin{bmatrix} 2 & 2 & -1 \\ 0 & 1 & 1 \\ 2 & 1 & 2 \end{bmatrix}\right)$$

$$\Delta_1 = 8,\ x_1 = 1$$

$$\Delta_2 = \det\left(\begin{bmatrix} -3 & 2 & -1 \\ -1 & 0 & 1 \\ 2 & 2 & 2 \end{bmatrix}\right)$$

$$\Delta_2 = 16,\ x_2 = 2$$

$$\Delta_3 = \det\left(\begin{bmatrix} -3 & 2 & 2 \\ -1 & 1 & 0 \\ 2 & 1 & 2 \end{bmatrix}\right)$$

$$\Delta_3 = -8,\ x_3 = -1$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$$

Тут також слід зазначити, що при зміні елементів матриць A та B, або взагалі їх розмірностей, при роботі в середовищі Maple натискання на клавішу Enter автоматично приведе до зміни проміжних і кінцевого результату.

1.5.3 Метод Гаусса

За допомогою бібліотеки **linalg** можна звести матрицю до різних спеціальних форм. Команда **gausselim**(A) використовується для зведення до трикутного вигляду. Алгоритм гауссова виключення без ділення реалізується функцією **ffgausselim**(A). Останню команду зручно використовувати для перевірки правильності проміжних обчислень при розв'язанні системи лінійних рівнянь **методом Гаусса** із застосуванням схеми "єдиного ділення"

```
> A:=matrix(3,3,[[-3,2,-1],[-1,1,1],[2,1,2]]);
B:=matrix([[2],[0],[2]]);
`Розширена матриця`=concat(A,B);
`Прямий хід`=ffgausselim(concat(A,B));
```

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

$$B := \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix}$$

$$\textit{Розширена матриця} = \begin{bmatrix} -3 & 2 & -1 & 2 \\ -1 & 1 & 1 & 0 \\ 2 & 1 & 2 & 2 \end{bmatrix}$$

$$\textit{Прямий хід} = \begin{bmatrix} -3 & 2 & -1 & 2 \\ 0 & -1 & -4 & 2 \\ 0 & 0 & 8 & -8 \end{bmatrix}$$

Команда **concat**(A1..,A2) була використана для склеювання декількох матриць горизонтально.

До трикутного вигляду звести матрицю можна так само за допомогою алгоритму Гаусса-Жордана - **gaussjord**.

```
> Y:=gaussjord(concat(A,B));
```

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

Останній стовпець матриці є розв’язком системи:

```
> X:=col(Y,4):
```

Подання вектора X у вигляді вектора-стовпця

```
X:=convert(X,Vector):'X'=LinearAlgebra[Transpose] (X);
x:='x':x[3]=X[3];
```

$$X = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$$

$$x_3 = -1$$

Правильність розв’язання можна перевірити перемноженням матриць

```
> evalm(A&*X);
```

$$[2, 0, 2]$$

Ми отримали вектор вільних членів.

В даному випадку, як і в більшості інших, команди Maple дозволяють дістати тільки кінцевий результат. Для отримання всього ходу розв'язання системи лінійних рівнянь методом виключення Гаусса необхідно скласти процедуру.

Розв'язання системи лінійних рівнянь методом виключення Гаусса

```
>gauss_solve_print:=proc(a::matrix,b::vector,step::boo
lean)
  local  i,j,k,c,r,x,m;
if  `linalg/rowdim`(a) <> `linalg/vectdim`(b) then
    error("Кількість стовпців матриці %1 повинна
дорівнювати розмірності вектора %2",a,b);
elif `linalg/rowdim`(a) <> `linalg/coldim`(a) then
    error("Матриця %1 повинна бути квадратною",a);
end if;
m:=`linalg/rowdim`(a);
c:=a;
c:=evalm(`linalg/augment`(c,b));
x:=vector(m);
if step then print(`Прямий хід методу виключення
Гаусса`) end if;
# Прямий хід
     for i from 1 to m do
        r:=1/c[i,i];
        c:=`linalg/mulrow`(c,i,r);#Ділення коефіцієнтів i-го
рядка на коефіцієнт cii
        if i <> m then
           for j from i+1  to m do
               r:=-c[j,i];
               c:=`linalg/addrow`(c,i,j,r);#Додавання до
елементів j-го рядка елементів i-го рядка, помножених на -c[j,i]
           end do;
        end if;
        if step then print(`Шаг`||i=eval(c)) end
if;#Виведення на екран монітора
     end do;
# Зворотний хід
     x[m]:=c[m,m+1];
     for i from m-1 to 1 by -1 do
          x[i]:= c[i,m+1]-sum(x[k]*c[i,k],k=(i+1)..m);
     end do;
     eval(x);
end proc:
```

Результати побудови розв'язання нашої задачі з друкуванням проміжних результатів подані нижче:

```
> B:=vector([2,0,2]):
sol:=gauss_solve_print(A,B,true);
```

Прямий хід методу виключення Гаусса

$$Крок1 = \begin{bmatrix} 1 & \frac{-2}{3} & \frac{1}{3} & \frac{-2}{3} \\ 0 & \frac{1}{3} & \frac{4}{3} & \frac{-2}{3} \\ 0 & \frac{7}{3} & \frac{4}{3} & \frac{10}{3} \end{bmatrix}$$

$$Крок2 = \begin{bmatrix} 1 & \frac{-2}{3} & \frac{1}{3} & \frac{-2}{3} \\ 0 & 1 & 4 & -2 \\ 0 & 0 & -8 & 8 \end{bmatrix}$$

$$Крок3 = \begin{bmatrix} 1 & \frac{-2}{3} & \frac{1}{3} & \frac{-2}{3} \\ 0 & 1 & 4 & -2 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

$$sol := [1, 2, -1]$$

В прямому ході методу розширена матриця системи **c** приводиться до верхнього трикутного вигляду з одиницями на головній діагоналі. При діленні коефіцієнтів *i-го* рядка на коефіцієнт c_{ii} дістаємо одиниці на головній діагоналі. Додаванням до елементів *j-го* рядка елементів *i-го* рядка, помножених на $(-c_{ji})$, обнуляється коефіцієнт c_{ji}.

Слід відмітити, що в процедурі передбачено виведення на екран монітора тільки основних проміжних результатів прямого ходу.

Відмітимо також, що процедура працездатна не у всіх випадках. Зокрема, треба стежити за тим, щоб коефіцієнт a_{ii} не був рівний нулю (цього легко домогтися перестановкою рядків вихідної системи лінійних рівнянь). Для програмного усунення подібних ситуацій процедуру треба дороблятиі. Однак в більшості випадків будуть отримані необхідні результати.

В процедурі *My_gauss* реалізована "*схема єдиного ділення*". Причому на *k*-ому кроці прямого ходу за провідний елемент приймається елемент c_{kk}. Як і попередня, ця процедура також працездатна не у всіх випадках.

```
> My_gauss:=proc(a::matrix,b::vector,step::boolean)
  local  i,j,k,c,x,m,gInt,step1::boolean;
step1:=false:
```

```
if  `linalg/rowdim`(a) <> `linalg/vectdim`(b)  then
    error("Кількість стовбців матриці %1 повинно
дорівнювати розмірності вектора %2",a,b);
elif `linalg/rowdim`(a) <> `linalg/coldim`(a) then
    error("Матриця %1 повинна бути квадратною",a);
end if;
m:=`linalg/rowdim`(a);
c:=a;
c:=evalm(`linalg/augment`(c,b));#Створення розширеної
матриці
x:=vector(m);
if step then print(`Прямий хід методу виключення
Гаусса - схема єдиного деления`) end if;
#  Прямий хід
for k to m-1 do #k-ий крок прямого ходу
     for i from k+1 to m do #Цикл по рядках
           for j from k+1 to m+1 do #Цикл по стовпцях
               c[i,j]:=c[k,k]*c[i,j]-
c[i,k]*c[k,j];#Основна формула
           end do;
           c[i,k]:=0;#Обнулення нижньої трикутної матриці
     end do;
    if step then
        print(`Крок`||k=eval(c)):#Виведення на екран
монітора
        #Перевірка можливості скорочення обох частин рівняння на ціле
число, більше 1, без отримання дробових коефіцієнтів
        for i from k+1 to m do #Цикл по рядках
        #Визначення найбільшого спільного дільника для рядка
розширеної матриці

gInt:=igcd(op(convert(`linalg/row`(c,i),list))):
           if gInt<>1 then
               #Перевірка на від'ємність елементів рядка

select(type,convert(`linalg/row`(c,i),list),positive);
               if nops(%)=0 then gInt:=-gInt end if;
               step1:=true:
               for j from k+1 to m+1 do #Цикл по стовпцях
                   c[i,j]:=c[i,j]/gInt;
               end do;
           end if:
        end do:
        if step1 then
```

```
            print(`Крок`||k||"."||1=eval(c)):
            step1:=false:
        end if:
    end if;
end do;
```
#Зворотний хід
```
x:=vector(m,0);k:='k':
     for i from m to 1 by -1 do
          x[i]:= (c[i,m+1]-
sum(x[k]*c[i,k],'k'=1..m))/c[i,i];
     end do;
     eval(x);
end proc:
```

- #Приклад 1

```
A:=matrix(3,3,[[-3,2,-1],[-
1,1,1],[2,1,2]]):B:=vector([2,0,2]):
sol:=My_gauss(A,B,true);
```

Прямий хід методу виключення Гаусса - схема єдиного ділення

$$Крок1 = \begin{bmatrix} -3 & 2 & -1 & 2 \\ 0 & -1 & -4 & 2 \\ 0 & -7 & -4 & -10 \end{bmatrix}$$

$$Крок2 = \begin{bmatrix} -3 & 2 & -1 & 2 \\ 0 & -1 & -4 & 2 \\ 0 & 0 & -24 & 24 \end{bmatrix}$$

$$Крок2.1 = \begin{bmatrix} -3 & 2 & -1 & 2 \\ 0 & -1 & -4 & 2 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

$$sol := [\,1,\ 2,\ -1\,]$$

Матриця Крок2.1 відрізняється від матриці Крок2 тільки тим, що елементи третього рядка поділені на число 24.

> #Приклад 2

```
A:=matrix(4,4,[[2,-1,-2,-3],[0,2,3,-2],[3,2,-1,2],[2,-
2,0,-1]]);B:=vector([8,5,4,0]):
sol:=My_gauss(A,B,true);
```

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

Прямий хід методу виключення Гаусса - схема єдиного ділення

$$Крок1 = \begin{bmatrix} 2 & -1 & -2 & -3 & 8 \\ 0 & 4 & 6 & -4 & 10 \\ 0 & 7 & 4 & 13 & -16 \\ 0 & -2 & 4 & 4 & -16 \end{bmatrix}$$

$$Крок1.1 = \begin{bmatrix} 2 & -1 & -2 & -3 & 8 \\ 0 & 2 & 3 & -2 & 5 \\ 0 & 7 & 4 & 13 & -16 \\ 0 & -1 & 2 & 2 & -8 \end{bmatrix}$$

$$Крок2 = \begin{bmatrix} 2 & -1 & -2 & -3 & 8 \\ 0 & 2 & 3 & -2 & 5 \\ 0 & 0 & -13 & 40 & -67 \\ 0 & 0 & 7 & 2 & -11 \end{bmatrix}$$

$$Крок3 = \begin{bmatrix} 2 & -1 & -2 & -3 & 8 \\ 0 & 2 & 3 & -2 & 5 \\ 0 & 0 & -13 & 40 & -67 \\ 0 & 0 & 0 & -306 & 612 \end{bmatrix}$$

$$Крок3.1 = \begin{bmatrix} 2 & -1 & -2 & -3 & 8 \\ 0 & 2 & 3 & -2 & 5 \\ 0 & 0 & -13 & 40 & -67 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

$$sol := [1, 2, -1, -2]$$

Матриця Крок3.1 відрізняється від матриці Крок3 тим, що елементи четвертого рядка поділені на число 306.

1.6 Векторна алгебра

1.6.1 Складання векторів

Для більш зручного зображення векторів та результатів дій над ними будемо задавати вектори точками їх початку та кінця. А координати точок будемо задавати координатами їх радіусів-векторів.

```
> restart:
pA:=vector(3);pB:=vector(3);pC:=vector(3);
```

```
pD:=vector(3);
```

$$pA := \text{array}(1 .. 3, [\ ])$$

$$pB := \text{array}(1 .. 3, [\ ])$$

$$pC := \text{array}(1 .. 3, [\ ])$$

$$pD := \text{array}(1 .. 3, [\ ])$$

```
> vectorAB:=evalm(pB-pA);vectorCD:=evalm(pD-pC);
```

$$vectorAB := [pB_1 - pA_1, pB_2 - pA_2, pB_3 - pA_3]$$

$$vectorCD := [pD_1 - pC_1, pD_2 - pC_2, pD_3 - pC_3]$$

Для зображення цих векторів потрібно задати конкретні числові значення координат точок A, B, C, D:

```
> pA:=vector(3,0);pB:=vector(3,[0,0,1]);
pC:=vector(3,[1,0,0]);
pD:=vector(3,[1,1,0]);vectorAB:=evalm(pB-pA);
vectorCD:=evalm(pD-pC);op(pC);
```

$$pA := [0, 0, 0]$$

$$pB := [0, 0, 1]$$

$$pC := [1, 0, 0]$$

$$pD := [1, 1, 0]$$

$$vectorAB := [0, 0, 1]$$

$$vectorCD := [0, 1, 0]$$

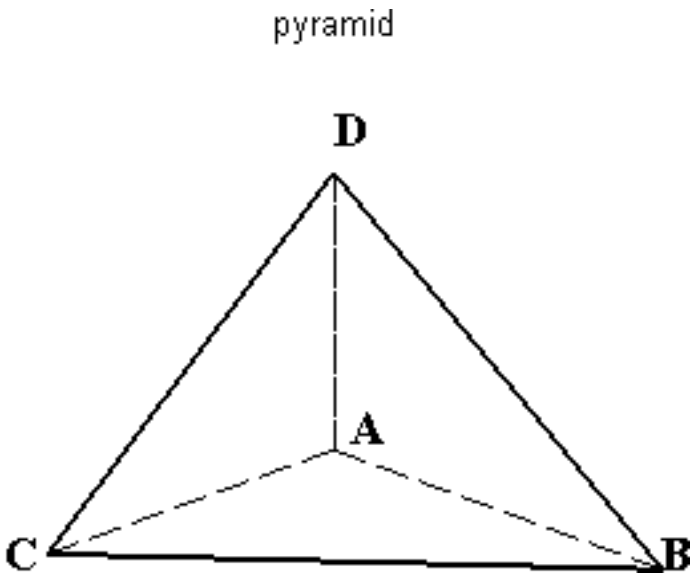
$$[1, 0, 0]$$

Для графічного зображення векторів потрібно підключити додаткові бібліотеки: **plots** та **plottools**

```
> with(plots):with(plottools):
```

Приклад.

Дана трикутна піраміда ABCD. Знайти суму векторів **AB**+**CD**+**AC**+**BC**+**DA**.



Розв'язання. Подамо вершини піраміди у вигляді тривимірних векторів

```
> pA:=vector(3);pB:=vector(3);pC:=vector(3);pD:=vector(3);
```

$$pA := \text{array}(1 .. 3, [\ ])$$

$$pB := \text{array}(1 .. 3, [\ ])$$

$$pC := \text{array}(1 .. 3, [\ ])$$

$$pD := \text{array}(1 .. 3, [\ ])$$

Знайдемо координати векторів **AB**, **CD**, **AC**, **BC**, **DA**:

```
> AB:=evalm(pB-pA);CD:=evalm(pD-pC);AC:=evalm(pC-pA);
BC:=evalm(pC-pB);DA:=evalm(pA-pD);
```

$$AB := [pB_1 - pA_1, pB_2 - pA_2, pB_3 - pA_3]$$

$$CD := [pD_1 - pC_1, pD_2 - pC_2, pD_3 - pC_3]$$

$$AC := [pC_1 - pA_1, pC_2 - pA_2, pC_3 - pA_3]$$

$$BC := [pC_1 - pB_1, pC_2 - pB_2, pC_3 - pB_3]$$

$$DA := [pA_1 - pD_1, pA_2 - pD_2, pA_3 - pD_3]$$

Тепер ми можемо знайти шукану суму

```
> AB+CD+AC+BC+DA:'AB+CD+AC+BC+DA'=evalm(%);
```

$$AB + CD + AC + BC + DA = [pC_1 - pA_1, pC_2 - pA_2, pC_3 - pA_3]$$

Очевидно, що ми отримали вектор **AC.** Цей результат можна отримати, скориставшись переставною та сполучною властивостями дії додавання векторів:

AB+CD+AC+BC+DA=(AB+BC+CD+DA)+AC=0+AC=AC.

Легко перевірити, що **AB+BC+CD+DA=0:**

```
> evalm(AB+BC+CD+DA);
```

$$[0, 0, 0]$$

Інакше і бути не може, адже початок і кінець підсумкового вектора збігаються.

1.6.2 Скалярний, векторний і мішаний добуток векторів

Скалярний добуток двох векторів

```
> a:=vector([5,6,3]):b:=vector([-5,0,3]):dotprod(a, b);
```

-16

Векторний добуток двох векторів

```
> d:=crossprod(a, b);
```

$$d := [18, -30, 30]$$

Мішаний добуток трьох векторів **a**, **b**, **c** можна визначити таким чином:

```
> a:=vector([5,6,3]);b:=vector([-
5,0,3]);c:=vector([1,2,-1]);
`abc`:=stackmatrix(a,b,c)=det(stackmatrix(a,b,c));#Міш
аний добуток
```

$$a := [5, 6, 3]$$

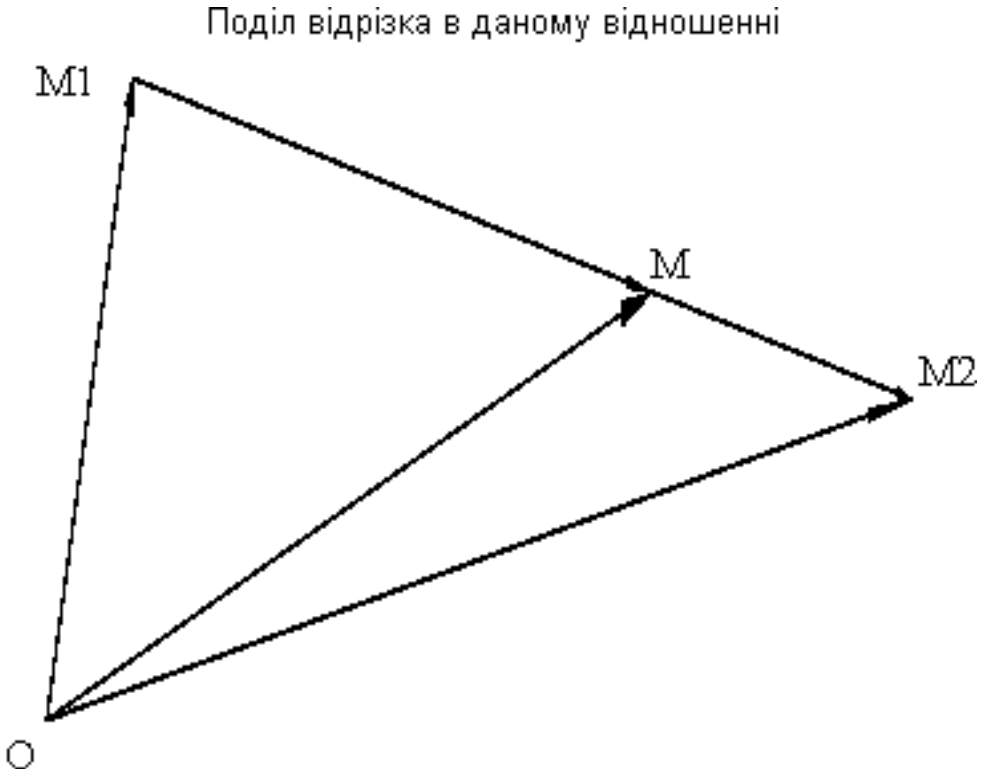
$$b := [-5, 0, 3]$$

$$c := [1, 2, -1]$$

$$abc := \begin{bmatrix} 5 & 6 & 3 \\ -5 & 0 & 3 \\ 1 & 2 & -1 \end{bmatrix} = -72$$

1.6.3 Поділ відрізка в даному відношенні

Знаючи координати точок M1(x1,y1) і M2(x2,y2) та відношення λ (λ=M1M/MM2), в якому точка M поділяє напрямлений відрізок M1M2, знайти координати точки M. Дивись рисунок.



Розв`язання.

Через x, y позначимо координати точки M. Нехай O - початок координат. Утворимо вектори: **OM1**, **OM2**, **OM**, **M1M**, **MM2**. Оскільки **M1M**=**OM**-**OM1**, а **MM2**=**OM2**-**OM**, то за умовою задачі **OM**-**OM1**= λ ***(OM2**-**OM)**. Звідси, користуючись умовами колінеарності векторів **M1M**, **MM2**, матимемо: x-$x1$= λ *($x2$-x); y-$y1$= λ *($y2$-y), або x=($x1$+ λ *$x2$)/(1+ λ); y=($y1$+ λ *$y2$)/(1+ λ).

Складемо програму, яка обчислює координати точки M і, головне, будує рисунок, що ілюструє результати розв`язання задачі:

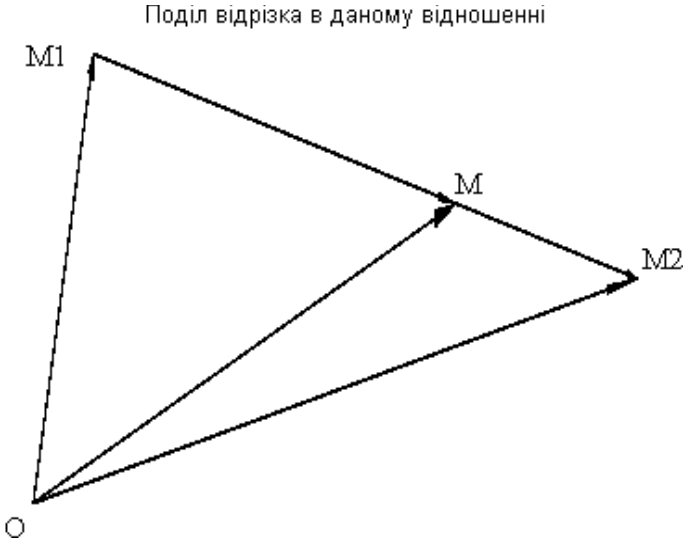
```
> restart:with(plots):with(plottools):
#Вихідні дані: OM1 та OM2 - координати точок кінців відрізка M1M2
OM1:=vector([1,2]):
OM2:=vector([10,1]):
```

#Знак додатний, якщо точка M знаходиться в середині відрізка M1M2, і від`ємний у протилежному випадку

lambda:=2:#Відношення довжин векторів λ=M1M/MM2 (-infinity< λ <-1,-1< λ <infinity)

```
wb:=.01:wh:=.04:hh:=.05:
OM:=evalm((OM1+lambda*OM2)/(1+lambda));#Координати точки
M
M1M:=evalm(OM-OM1):MM2:=evalm(OM2-OM):
LM1M:=arrow(OM1, M1M, wb, wh, hh,
color=black):LM1:=arrow([0,0],OM1, wb, .08, hh,
color=black):LM:=arrow([0,0],OM, wb, .08, hh,
color=black):
LM2:=arrow([0,0], OM2, wb, wh, hh, color=black):
LMM2:=arrow(OM, MM2, wb, wh, hh, color=black):
gp1:=PLOT(TEXT([-.3,-
.1],'O',FONT(SYMBOL,12)),TEXT([OM1[1]-
.4,2],'M1',ALIGNLEFT,FONT(SYMBOL,12)),TEXT([evalf(OM[1
]),evalf(OM[2])+.1],'M',ALIGNRIGHT,FONT(SYMBOL,12)),TE
XT([OM2[1]+.1,OM2[2]+.1],'M2',ALIGNRIGHT,FONT(SYMBOL,1
2))):
plots[display](LM1,LM1M,LM,LM2,LMM2,gp1,title="Поділ
відрізка в даному відношенні", axes=none);
Warning, the name changecoords has been redefined
```

$$OM := \left[7, \frac{4}{3}\right]$$



Для зручності користування програмою її краще оформити у вигляді процедури.

Поділ відрізка в даному відношенні: M1(x1,y1) і M2(x2,y2) - координати точок кінців відрізка M1M; λ=M1M/MM2

```
> restart:
Segmentation := proc (M1::vector,M2::vector,
lambda,illustration::boolean)#Поділ відрізка в даному
відношенні
local wb,wh,hh,OM,M1M,MM2,LM1M,LM1,LM,LM2,LMM2,gp1:
wb:=.01:wh:=.04:hh:=.05:
OM:=evalm((M1+lambda*M2)/(1+lambda)):
M1M:=evalm(OM-M1):MM2:=evalm(M2-OM):
LM1M:=plottools['arrow'](M1, M1M, wb, wh, hh,
color=black):LM1:=plottools['arrow']([0,0],M1, wb,
.08, hh, color=black):LM:=plottools['arrow']([0,0],OM,
wb, .08, hh, color=black):
LM2:=plottools['arrow']([0,0], M2, wb, wh, hh,
color=black):
LMM2:=plottools['arrow'](OM, MM2, wb, wh, hh,
color=black):
gp1:=PLOT(TEXT([-.3,-
.1],'O',FONT(SYMBOL,12)),TEXT([M1[1]-
.4,2],'M1',ALIGNLEFT,FONT(SYMBOL,12)),TEXT([evalf(OM[1
]),evalf(OM[2])+.15],'M',ALIGNLEFT,FONT(SYMBOL,12)),TE
XT([M2[1]+.1,M2[2]+.1],'M2',ALIGNRIGHT,FONT(SYMBOL,12)
)):
print(`Координати точки M`=evalm(OM));
print(`Координати точок: M1`=evalm(M1),`
M2`=evalm(M2),` lambda`=lambda);
if illustration then

plots['display'](LM1,LM1M,LM,LM2,LMM2,gp1,title="Поділ
відрізка в даному відношенні", axes=none)
end if:
end proc;
```

Segmentation := proc(*M1*::*vector*, *M2*::*vector*, λ, *illustration*::*boolean*)

local *wb*, *wh*, *hh*, *OM*, *M1M*, *MM2*, *LM1M*, *LM1*, *LM*, *LM2*, *LMM2*, *gp1*;

wb := .01;

wh := .04;

hh := .05;

OM := evalm((*M1* + λ×*M2*)/(1 + λ));

M1M := evalm(*OM* – *M1*);

MM2 := evalm(*M2* – *OM*);

LM1M := *plottools*['*arrow*'](*M1*, *M1M*, *wb*, *wh*, *hh*, *color* = *black*);

LM1 := *plottools*['*arrow*']([0, 0], *M1*, *wb*, .08, *hh*, *color* = *black*);

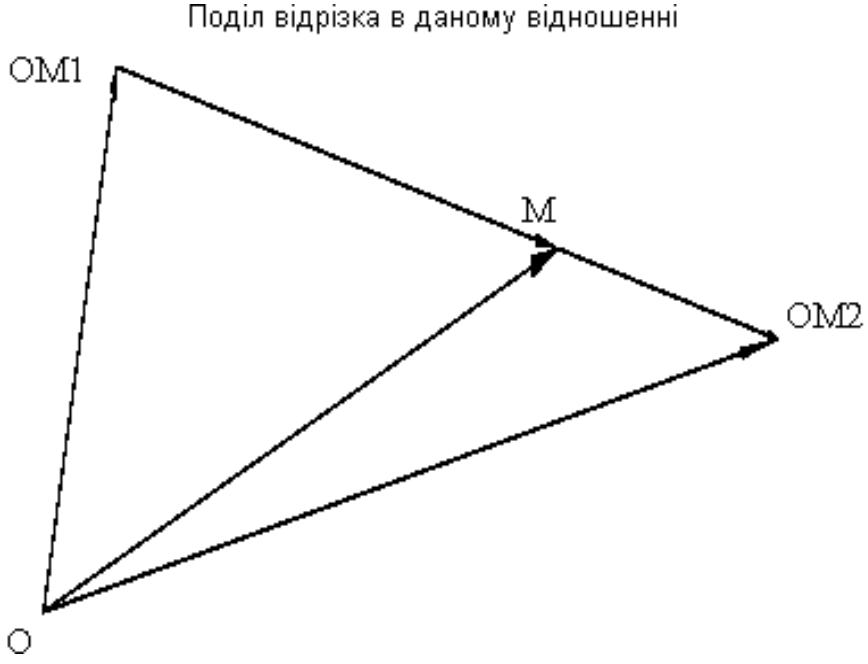
```
LM := plottools['arrow']([0, 0], OM, wb, .08, hh, color = black);
LM2 := plottools['arrow']([0, 0], M2, wb, wh, hh, color = black);
LMM2 := plottools['arrow'](OM, MM2, wb, wh, hh, color = black);
```

Зформуємо вихідні дані та викличемо процедуру:

```
> OM1:=vector([1,2]):
OM2:=vector([10,1]):
lambda:=2:
Segmentation(OM1,OM2,lambda,true);
```

$$Координати\ точки\ M = \left[7, \frac{4}{3}\right]$$

$$Координати\ точок:\ M1 = [1, 2],\ M2 = [10, 1],\ lambda = 2$$



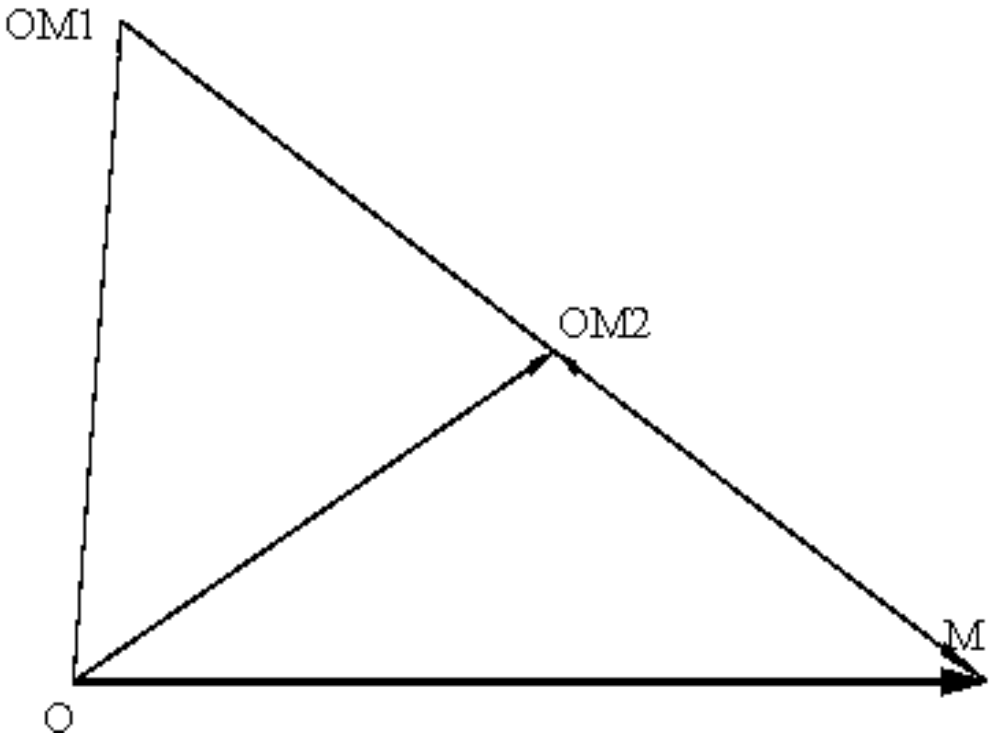
Випадок, коли від`ємне:

```
> OM1:=vector([1,2]):
OM2:=vector([10,1]):
lambda:=-2:
Segmentation(OM1,OM2,lambda,true);
```

Координати точки M= [19, 0]

Координати точок: M1= [1, 2], *M2* = [10, 1], *lambda* = -2

Поділ відрізка в даному відношенні



2. Аналітична геометрія

2.1 Аналітична геометрія на площині

2.1.1 Рівняння першого степеня з двома змінними (пряма на площині)

2.1.1.1 Рівняння прямої з кутовим коефіцієнтом

Рівняння прямої з кутовим коефіцієнтом має вигляд $y=k*x+b$.

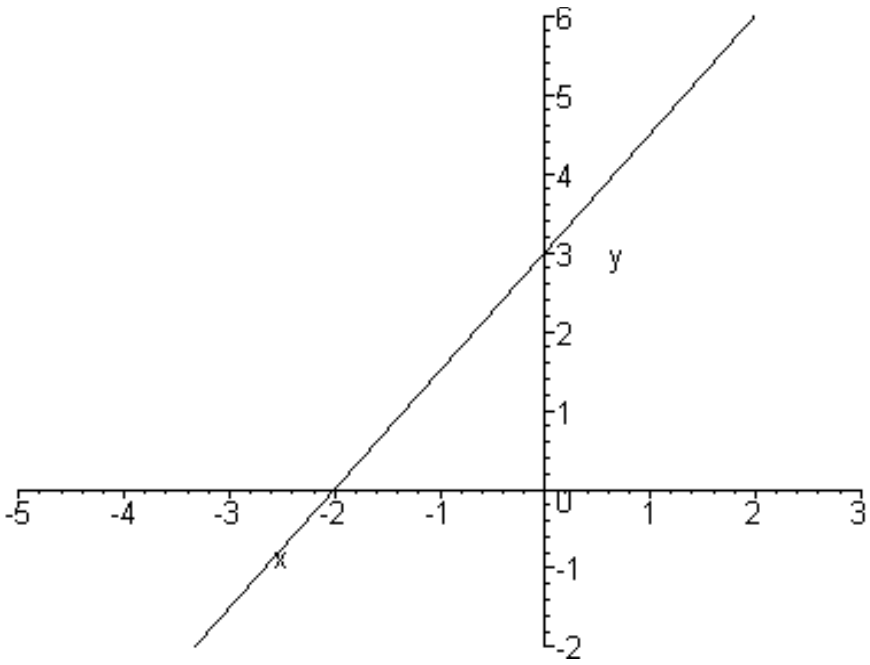
```
> restart:
l1:=y=k*x+b;
```

$$l1 := y = k\,x + b$$

Для побудови графіка прямої необхідно задати конкретні числові значення кутового коефіцієнта **k** та величини ***b*** - ординати точки перетину прямої **l1** з вертикальною віссю координат

```
> k:=1.5:b:=3:
print(`Графік прямої    y`=k*x+b);
plot(rhs(l1),x=-5..3,y=-2..6);
```

$$Графік\ прямої \quad y = 1.5\,x + 3$$



Знайдемо *k* з рівняння прямої

```
> k:='k':b:='b':k:=solve(l1,k);
```

$$k := \frac{y-b}{x}$$

Звідси випливає геометричний зміст кутового коефіцієнта *k*, що дорівнює *tg(*ϕ*)*, ϕ - кут, утворений прямою *l1* з віссю *Ox*.

Значення коефіцієнта k та кута ϕ можна обчислити, підставивши в формулу координати будь-якої точки прямої *l1* **(***x<>0)*

```
> k:=subs(x=-2,y=0,b=3,k);phi:=arctan(k);
```

$$k := \frac{3}{2}$$

$$\phi := \arctan\left(\frac{3}{2}\right)$$

Подамо останній результат у вигляді десяткового дробу

```
> evalf(%);
```

$$.9827937232$$

Конвертування радіан в градуси

```
> convert(%,degrees);
```

$$176.9028702 \frac{degrees}{p}$$

```
> evalf(%);
```

$$56.30993246 degrees$$

2.1.1.2 Рівняння прямої, що проходить через дану точку в даному напрямі

Запишемо рівняння прямої з кутовим коефіцієнтом

```
> restart:l1:=y=k*x+b:l1;
```

$$y = k\,x + b$$

Координати будь-якої точки M0(**x**0,**y**0), що належить прямій *l1*, задамо у вигляді вектора

```
> M0:=[x0,y0];
```

$$M0 := [x0, y0]$$

Оскільки за умовою точка M0 належить прямій *l1*, то координати цієї точки задовольняють рівняння прямої

```
> eq1:=subs(x=M0[1],y=M0[2],l1):eq1;
```

$$y0 = k\,x0 + b$$

Віднімемо здобуте рівняння від рівняння прямої *l1*

```
> l2:=l1-eq1:l2;
```

$$y - y0 = k\,x - k\,x0$$

Ми дістали рівняння прямої, що проходить через дану точку в даному напрямі. Надамо цьому рівнянню остаточного вигляду

```
> l2:=collect(l2,k):l2;
```

$$y - y0 = (x - x0)\,k$$

Рівняння прямої з кутовим коефіцієнтом

```
> k:=-3/2:b:=3:l1;
```

$$y = -\frac{3}{2}x + 3$$

неважко подати у вигляді рівняння прямої, що проходить через дану точку в даному напрямі. Для цього потрібно надати числові значення координатам (x0, y0):

```
> M0[1]:=0;M0[2]:=subs(x=M0[1],rhs(l1));
```

$$M0_1 := 0$$

$$M0_2 := 3$$

Підставимо отримані значення в рівняння прямої *l2*

```
> l2a:=subs(x0=M0[1],y0=M0[2],l2):l2a;
```

$$y - 3 = -\frac{3}{2}x$$

або

```
> l2a:=y=solve(l2a,y):l2a;
```

$$y = -\frac{3}{2}x + 3$$

Ми отримали те саме рівняння *l1*, що можна перевірити і так:

```
> l2a-l1;
```

$$0 = 0$$

Неважко переконатися, що той самий результат ми дістанемо, якщо в якості точки M0 візьмемо не точку (0,3), а будь-яку іншу точку, що належить прямій *l1*. Для цього потрібно задати будь-яке інше значення координати **x**0, наприклад, **x**0=4:

```
> M0[1]:=4:
M0[2]:=subs(x=M0[1],rhs(l1));
l2a:=subs(x0=M0[1],y0=M0[2],l2):l2a:=y=solve(l2a,y):so
rt(rhs(l2a)):l2a;l2a-l1:
```

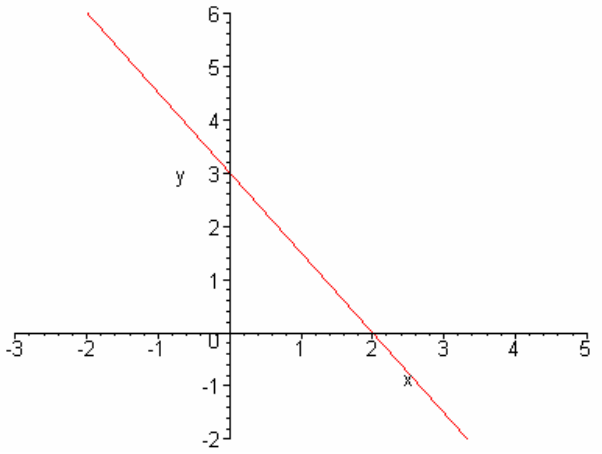
$$M0_2 := -3$$

$$y = -\frac{3}{2}x + 3$$

Графік розглянутої прямої має вигляд

```
> kk:=coeff(rhs(l2a),x):bb:=coeff(rhs(l2a),x,0):
print(`Графік прямої    y`=kk*x+bb);
plot(rhs(l2a),x=-3..5,y=-2..6);
```

$$Графік\ прямої \quad y = -\frac{3}{2}x + 3$$



Як видно, два оператори перед командою print тут використані для визначення значень коефіцієнтів *k* та *b* безпосередньо з рівняння прямої *l2a*, для якої будується графік. Це дозволяє уникнути випадків невідповідності між виведеним рівнянням прямої та її графіком. Навіть якщо змінити значення коефіцієнтів *k* та *b* перед побудовою графіка прямої, вони все одно будуть переобчислені у відповідності з рівнянням прямої *l2a***.**

2.1.1.3 Рівняння прямої, що проходить через дві дані точки

Запишемо рівняння прямої**,** що проходить через дану точку в даному напрямі

```
> restart:
M1:=[x1,y1]:
l2:=y-M1[2]=k*(x-M1[1]):l2;
```

$$y - y1 = k\,(x - x1)$$

задамо координати будь-якої точки M2(**x**2,**y**2), що належить прямій *l2*

```
> M2:=[x2,y2];
```

$$M2 := [x2, y2]$$

Оскільки за умовою точка M2 належить прямій *l2*, то координати цієї точки задовольняють рівняння прямої

```
> eq2:=subs(x=M2[1],y=M2[2],l2):eq2;
```

$$y2 - y1 = k\,(x2 - x1)$$

Поділемо рівняння прямої *l2* почленно на здобуте рівняння

```
> l3:=lhs(l2)/lhs(eq2)=rhs(l2)/rhs(eq2):l3;
```

$$\frac{y - y1}{y2 - y1} = \frac{x - x1}{x2 - x1}$$

Ми здобули рівняння прямої, що проходить через дві дані точки M1(**x**1,**y**1), M2(**x**2,**y**2).

Рівняння прямої з кутовим коефіцієнтом

```
> k:=1:b:=-3:l1:=y=k*x+b:l1;
```

$$y = x - 3$$

подамо у вигляді рівняння прямої, що проходить через дві дані точки M1(**x**1,**y**1), M2(**x**2,**y**2), а потім знову перейдемо до початкового вигляду рівняння.

Задамо координати точок M1(**x**1,**y**1), M2(**x**2,**y**2):

```
> M1[1]:=0;M1[2]:=subs(x=M1[1],rhs(l1));
M2[1]:=3;M2[2]:=subs(x=M2[1],rhs(l1));
```

$$M1_1 := 0$$

$$M1_2 := -3$$

$$M2_1 := 3$$

$$M2_2 := 0$$

Підставимо отримані значення в рівняння прямої *l3*

```
> l3a:=subs(x1=M1[1],y1=M1[2],x2=M2[1],
  y2=M2[2],l3):l3a;
```

$$\frac{1}{3}y + 1 = \frac{1}{3}x$$

або

```
> l3a:=y=solve(l3a,y):l3a;
```

$$y = x - 3$$

Ми отримали те саме рівняння *l1*, що можна перевірити і так:

```
> l3a-l1;
```

$$0 = 0$$

Неважко переконатися, що той самий результат ми дістанемо, якщо в якості точок M1(**x**1,**y**1), M2(**x**2,**y**2) візьмемо будь-які інші точки, що належить прямій *l1*. Наприклад

```
> M1[1]:=-5;M1[2]:=subs(x=M1[1],rhs(l1));
M2[1]:=5;M2[2]:=subs(x=M2[1],rhs(l1));
l3a:=subs(x1=M1[1],y1=M1[2],x2=M2[1],y2=M2[2],l3):l3a;
l3a:=y=solve(l3a,y):sort(rhs(l3a)):l3a;
```

$$M1_1 := -5$$

$$M1_2 := -8$$

$$M2_1 := 5$$

$$M2_2 := 2$$

$$\frac{1}{10}y + \frac{4}{5} = \frac{1}{10}x + \frac{1}{2}$$

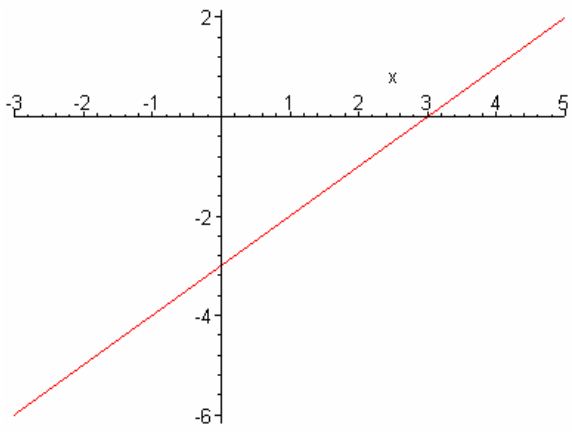
$$y = x - 3$$

Графік розглянутої прямої має вигляд

```
> kk:=coeff(rhs(l3a),x):bb:=coeff(rhs(l3a),x,0):
> print(`Графік прямої     y`=kk*x+bb);
```

```
plot(rhs(l3a),x=-3..5);
```

Графік прямої $y=x-3$



2.1.1.4 Загальне рівняння прямої на площині

У загальному вигляді рівняння прямої записується так: $Ax+By+C=0$.

Числа A, B, C та координати поточної точки прямої M(x,y) задамо у вигляді списків

```
> restart:
lineG_Eq:=A*x+B*y+C=0:lineG_Eq;
ABC:=([coeff(lhs(lineG_Eq),x),
coeff(lhs(lineG_Eq),y),op(nops(lhs(lineG_Eq)),lhs(lineG_Eq))]); pointM:=([x,y]);
```

$$A\,x + B\,y + C = 0$$

$$ABC := [A, B, C]$$

$$pointM := [x, y]$$

Доведемо таку теорему.

Теорема. Вектор **N**(A,B) перпендикулярний до прямої, що визначається рівнянням $Ax+By+C=0$.

Доведення. Нехай M1($x1$,$y1$) і M2($x2$,$y2$) деякі точки, що лежать на прямій $Ax+By+C=0$.

```
> M1 := [x1, y1]:M2 := [x2, y2]:
```

Це означає, що

```
> lineG_Eq1:=subs(x=M1[1],y=M1[2],lineG_Eq):lineG_Eq1;
lineG_Eq2:=subs(x=M2[1],y=M2[2],lineG_Eq):lineG_Eq2;
```

$$A\,x1 + B\,y1 + C = 0$$

$$A\,x2 + B\,y2 + C = 0$$

Віднімаючи від другої рівності першу, матимемо

```
> lineG_Eq3:=lineG_Eq2-
lineG_Eq1:collect(lineG_Eq3,{A,B});
```

$$(x2 - x1)\,A + (y2 - y1)\,B = 0$$

Остання рівність означає, що скалярний добуток векторів **N**(*A*,*B*) та **S**(*x*2-*x*1,*y*2-*y*1) дорівнює нулю, отже вектор **N**(*A*,*B*) перпендикулярний до вектора **S**(*x*2-*x*1,*y*2-*y*1). А оскільки вектор **S** лежить на прямій *A*x+*B*y+*C=0*, то теорему доведено.

Проілюструємо доведену властивість, побудувавши графік прямої *Ax+By+C=0* та вектора **N**(*A*,*B*):

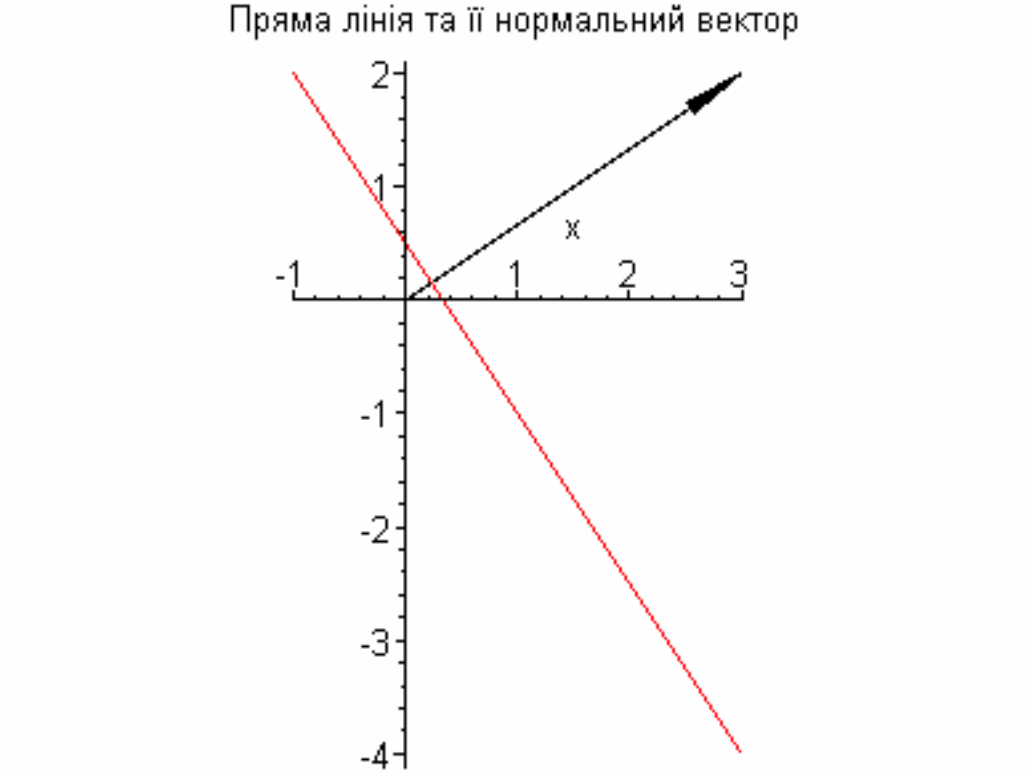
```
> lineG_EqP:=subs(A=3,B=2,C=-1,lineG_Eq):
print(`Загальне рівняння прямої`);lineG_EqP;
N_Line:=[coeff(lhs(lineG_EqP),x),coeff(lhs(lineG_EqP),
y)]:
lineG_EqP:=solve(lineG_EqP,y):
print(`Рівняння прямої з кутовим
коефіцієнтом`);y=lineG_EqP;
g1:=plot(lineG_EqP,x=-1..3,title="Пряма лінія та її
нормальний вектор"):
wb:=.01:wh:=.12:hh:=.15:
N_vect:=plottools['arrow']([0,0], N_Line, wb, wh, hh,
color=black):
plots['display'](g1,N_vect,title="Пряма лінія
(червоний колір) та її нормальний вектор",
axes=normal,scaling=CONSTRAINED);
```

Загальне рівняння прямої

$$3\,x + 2\,y - 1 = 0$$

Рівняння прямої з кутовим коефіцієнтом

$$y = -\frac{3}{2}x + \frac{1}{2}$$



2.1.1.5 Відстань від точки до прямої на площині

Нехай потрібно знайти відстань від деякої точки M0($x0$,$y0$) до прямої, що визначається рівнянням Ax+By+C=0, тобто знайти відстань d=|***MK***| (див. рис.).

```
> lineG_Eq:=A*x+B*y+C=0:lineG_Eq;
pointM0:=[x0,y0];
```

$$A\,x + B\,y + C = 0$$

$$pointM0 := [x0, y0]$$

Позначимо через K(xk,yk) координати точки K. Тоді вектор ***M0K*** дорівнює

```
> pointK:=[xk,yk]:M0K:=pointK-pointM0;
```

$$M0K := [-x0 + xk, -y0 + yk]$$

```
>
Nline:=[coeff(lhs(lineG_Eq),x),coeff(lhs(lineG_Eq),y)]
;
```

$$Nline := [A, B]$$

Але вектори ***M0K*** та ***N***(A,B) колінеарні. Тому модуль їх скалярного добутку дорівнює добутку модулів векторів ***M0K*** та ***N***(A,B): |***M0K***·**N**|=|***M0K***|·|***N***|. Звідки дістанемо

d=|***M0K***|=|***M0K***·**N**|/|***N***|=|Ax0+By0+C|/sqrt(A^2+B^2).

Складемо процедуру визначення відстані від точки M0($x0$,$y0$) до прямої, що визначається рівнянням $Ax+By+C=0$. Передбачимо обчислення точки перетину даної прямої та прямої, що проведена через т. M0 перпендикулярно до даної прямої, та побудову графіка, що ілюструє задані й знайдені елементи.

```
> restart:
#Визначення відстані від точки M0(x0,y0) до прямої Ax+By+C=0 з
візуалізацією точки та прямої
DistP_Line2d:= proc (M0::list,A,
B,C,illustration::boolean,Xmin,Xmax)
#illustration=true - виведення графіка, illustration=false - графік не
виводиться;
#Xmin, Xmax - область візуалізації графіка.
local
d,sols,g1,eq1,eq2,xk,yk,M1,M2,Xleft,Xright,Ybelow,Yabo
ve:
global K:
d:=abs(A*M0[1]+B*M0[2]+C)/sqrt(A^2+B^2):
M1:=[x1,y1]:M2:=[x2,y2]:
eq1:=A*x+B*y+C=0:
print(lhs(eq1)=`0 - рівняння прямої лінії`,`  точка
M0`=M0):
print(`Відстань  d`=d):
if coeff(lhs(eq1),x)<>0 then
       eq2:=y-M0[2]=(B/A)*(x-M0[1]):
else
       eq2:=x=M0[1]
end if;
sols:=solve({eq1,eq2},{x,y}):xk:=subs(sols,x);yk:=subs
(sols,y);
K:=[xk,yk]:
print(`Точка перетину  K`=K):
if B=0 then
   M1[2]:=yk-5:M2[2]:=yk+5:M1[1]:=xk:M2[1]:=xk
   #M1[2]:=min(M0[2],yk):M2[2]:=max(M0[2],yk):
   #if M1[2]=M2[2] then M1[2]:=yk-5:M2[2]:=yk+5: end
if:
elif A=0 then
   M1[1]:=xk-5:M2[1]:=xk+5:M1[2]:=yk:M2[2]:=yk
else
   M1[1]:=evalf(xk-.5*d): M2[1]:=evalf(xk+.5*d):
   M1[2]:=solve(subs(x=M1[1],eq1),y):
   M2[2]:=solve(subs(x=M2[1],eq1),y):
end
```

```
if;Xleft:=min(Xmin,M0[1],M1[1],M2[1],xk):Xright:=max(X
max,M0[1],M1[1],M2[1],xk):
Ybelow:=min(M0[2],M1[2],M2[2],yk):Yabove:=max(M0[2],M1
[2],M2[2],yk):
g1:=PLOT(POINTS(M0,SYMBOL(CIRCLE,10)),
POINTS([xk,yk],SYMBOL(CIRCLE,10)),
TEXT([(M0[1]+xk)*.5,(M0[2]+yk)*.5],"d",ALIGNBELOW,ALIG
NLEFT,FONT(TIMES, BOLD,14)),
TEXT(M0,"M0",ALIGNBELOW,ALIGNRIGHT,FONT(SYMBOL,14)),
TEXT(M1,"M1",ALIGNABOVE,ALIGNLEFT,FONT(SYMBOL,14)),
TEXT(M2,"M2",ALIGNABOVE,ALIGNRIGHT,FONT(SYMBOL,14)),
TEXT(K,"K",ALIGNABOVE,ALIGNRIGHT,FONT(SYMBOL,14)),
CURVES([M0,[xk,yk]],THICKNESS(3),LINESTYLE(1)),CURVES(
[M1,M2]),
AXESSTYLE(NORMAL),SCALING(CONSTRAINED),VIEW(Xleft..Xri
ght,Ybelow..Yabove)):
if illustration then
      plots['display']({g1},title="Відстань від точки
до прямої ")
end if:
end proc;
```

DistP_Line2d := **proc**(*M0*::*list*, *A*, *B*, *C*, *illustration*::*boolean*, *Xmin*, *Xmax*)
local *d*, *sols*, *g1*, *eq1*, *eq2*, *xk*, *yk*, *M1*, *M2*, *Xleft*, *Xright*, *Ybelow*, *Yabove*;
global *K*;
d := abs(*A*×*M0*[1] + *B*×*M0*[2] + *C*)/sqrt(*A*^2 + *B*^2);
M1 := [*x1*, *y1*];

M2 := [x2, y2];

eq1 := *A*×*x*+*B*×*y*+*C* = 0;

print(lhs(*eq1*) = `*0 - рівняння прямої лінії*`,` *точка M0*` = *M0*);

print(`*Відстань d*` = *d*);

if coeff(lhs(*eq1*),*x*) <> 0 **then** *eq2* := *y*-*M0*[2] = *B*×(x-M0[1])/A

else eq2 := x = M0[1]

end if;

sols := solve({eq1, eq2},{x, y});

xk := subs(sols,x);

yk := subs(sols,y);

K := [xk, yk];

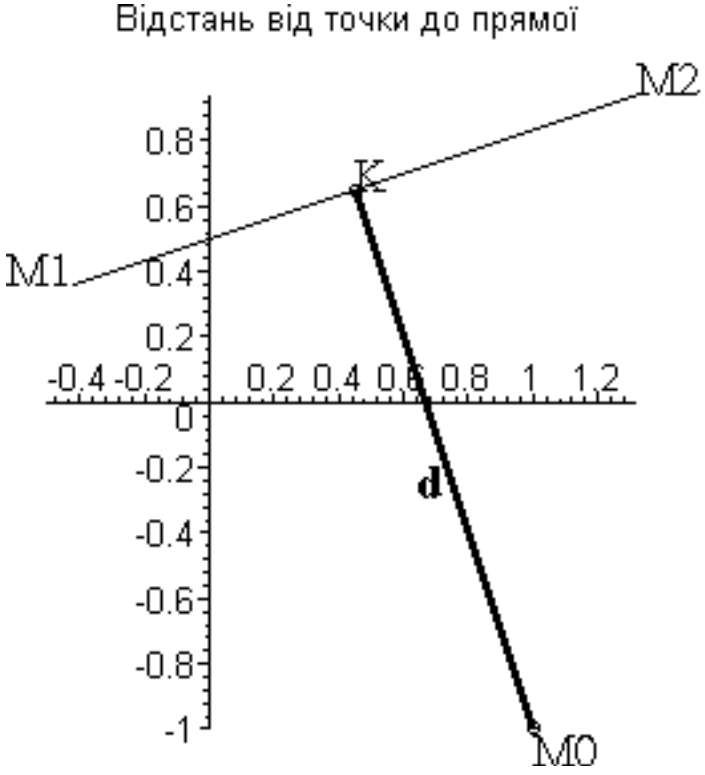
```
print(`Точка перетину  K` = K);
if B = 0 then M1[2] := yk-5; M2[2] := yk+5; M1[1] := xk; M2[1] := xk
elif A = 0 then M1[1] := xk-5; M2[1] := xk+5; M1[2] := yk; M2[2] := yk
else
        M1[1] := evalf(xk-.5×d);
        M2[1] := evalf(xk+.5×d);
        M1[2] := solve(subs(x = M1[1],eq1),y); M2[2] := solve(subs(x =
M2[1],eq1),y)
end if; Xleft := min(Xmin,M0[1],M1[1],M2[1],xk); Xright :=
max(Xmax,M0[1],M1[1],M2[1],xk); Ybelow :=
min(M0[2],M1[2],M2[2],yk); Yabove := max(M0[2],M1[2],M2[2],yk); g1
:= PLOT(POINTS(M0,SYMBOL(CIRCLE,10)),POINTS([xk,
yk],SYMBOL(CIRCLE,10)),TEXT([.5*M0[1]+.5×xk,
.5×M0[2]+.5×yk],"d",ALIGNBELOW,ALIGNLEFT,FONT(TIMES,BOL
D,14)),TEXT(M0,"M0",ALIGNBELOW,ALIGNRIGHT,FONT(SYMBO
L,14)),TEXT(M1,"M1",ALIGNABOVE,ALIGNLEFT,FONT(SYMBOL,1
4)),TEXT(M2,"M2",ALIGNABOVE,ALIGNRIGHT,FONT(SYMBOL,14
)),TEXT(K,"K",ALIGNABOVE,ALIGNRIGHT,FONT(SYMBOL,14)),C
URVES([M0, [xk, yk]],THICKNESS(3),LINESTYLE(1)),CURVES([M1,
M2]),AXESSTYLE(NORMAL),SCALING(CONSTRAINED),VIEW(Xle
ft .. Xright,Ybelow .. Yabove)); if illustration then
plots['display']({g1},title = "Відстань від точки до прямої ") end if end
proc
```

```
> MM0:=[1,-1]:A1:=-2:B1:=6:C1:=-3:
DistP_Line2d(MM0,A1,B1,C1,true,-.5,.5);
```

-2x+6y-3 = 0 - рівняння прямої лінії, точка M0 = [1, -1]

Відстань d = $\frac{11}{20}\sqrt{10}$

Точка перетину K = [9/20, 13/20]

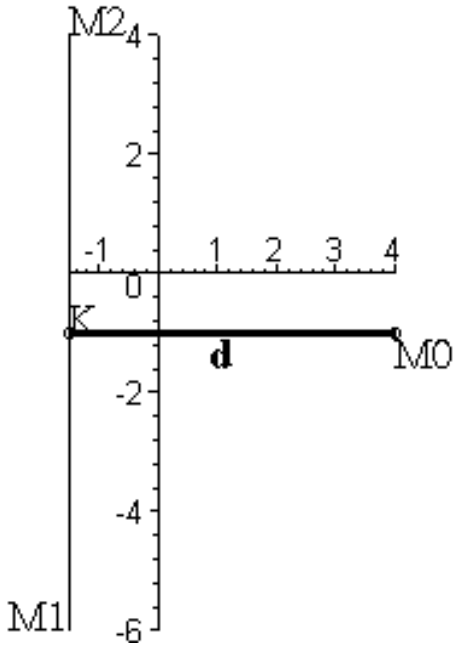



```
> MM0:=[4,-1]:A1:=-2:B1:=0:C1:=-3:
DistP_Line2d(MM0,A1,B1,C1,true,-.5,.5);
```

-2*x*-3 = 0 - рівняння прямої лінії, точка M0 = [4, -1]

Відстань d = $\frac{11}{2}$

Точка перетину K = [-3/2, -1]



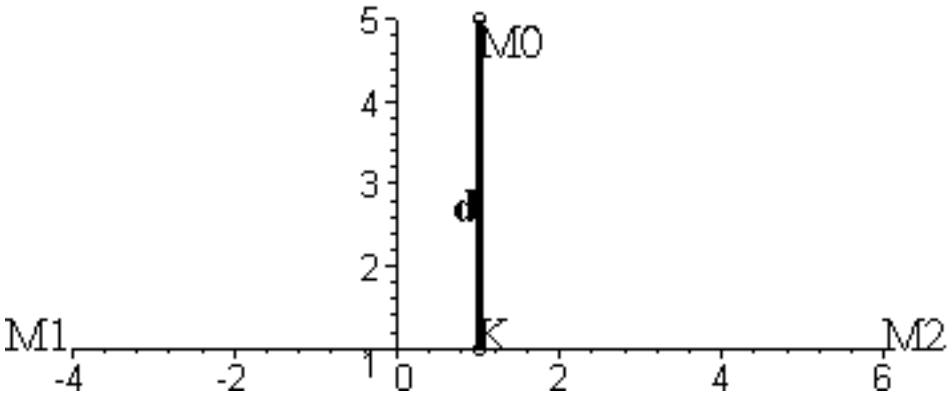
```
> MM0:=[1,5]:A1:=0:B1:=6:C1:=-6:
DistP_Line2d(MM0,A1,B1,C1,true,-.5,.5);
```

-6+6y = 0 - рівняння прямої лінії, точка M0 = [1, 5]

Відстань d = 4

Точка перетину K = [1, 1]

Відстань від точки до прямої



Приклад. Знайти висоту $|AK|$ трикутника ABC, якщо $A(2;3)$, $B(-1;2)$, $C(1;-1)$.
Розв'язання. Знайдемо висоту $|AK|$ як відстань від точки A до прямої, що проходить через сторону BC трикутника. Задамо точки A, B, C:

```
> A:=[2,3]:B:=[-1,2]:C:=[1,-1]:
```

Запишемо рівняння сторони BC як рівняння прямої, що проходить через дві задані точки B і C:

```
> LineBC:=(y-B[2])/(C[2]-B[2])=(x-B[1])/(C[1]-
B[1]):LineBC;
```

$$-\frac{1}{3}y+\frac{2}{3}=\frac{1}{2}x+\frac{1}{2}$$

Приведемо отримане рівняння до загального вигляду

```
> LineBCG:=rhs(LineBC)-lhs(LineBC)=0:
LineBCG:=sort(lhs(LineBCG),[x,y])=0:LineBCG;
op(nops(lhs(LineBCG)),lhs(LineBCG));
```

$$\frac{1}{2}x+\frac{1}{3}y-\frac{1}{6}=0$$

$$\frac{-1}{6}$$

Викличемо процедуру знаходження відстані від точки до прямої

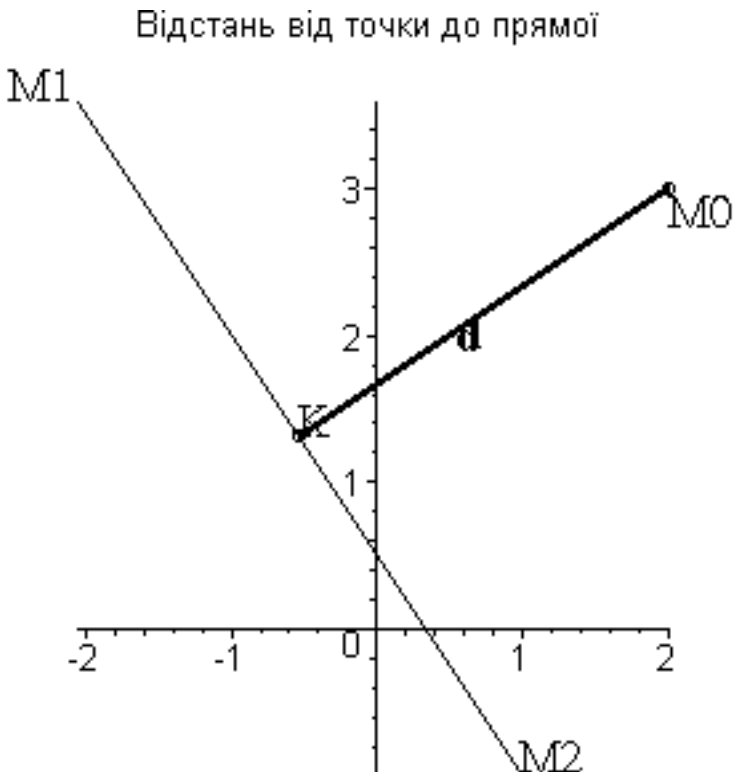
```
> MM0:=[1,-1]:A1:=-2:B1:=6:C1:=-3:lhsBC:=lhs(LineBCG);
DistP_Line2d(A,coeff(lhsBC,x),coeff(lhsBC,y),op(nops(l
hsBC),lhsBC),true,-.5,.5);
```

$$lhsBC := \frac{1}{2}x+\frac{1}{3}y-\frac{1}{6}$$

lhsBC $\frac{1}{2}$x+$\frac{1}{3}$y-$\frac{1}{6}$ = 0 - рівняння прямої лінії, точка M0 = [2, 3]

Відстань d = $\frac{11}{13}\sqrt{13}$

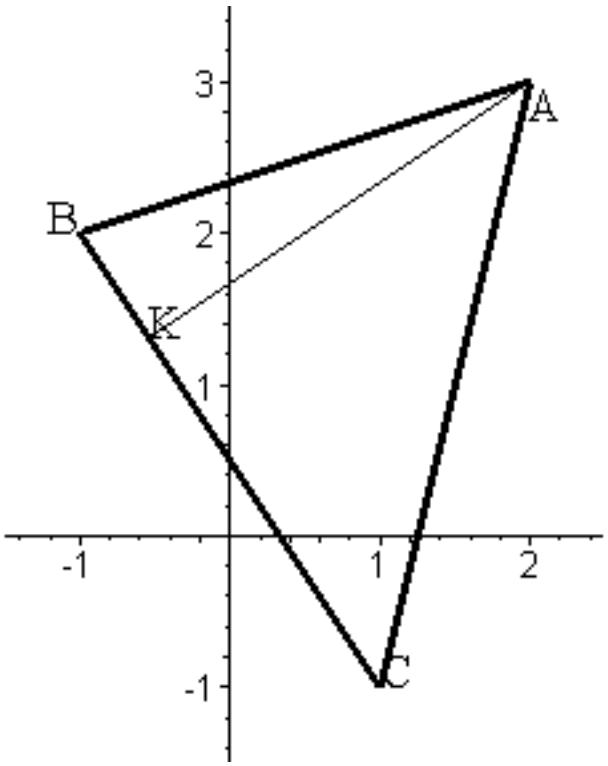
Точка перетину K = [-7/13, 17/13]



Побудуємо графік трикутника та його висоти

```
> PLOT(TEXT(A,"A",ALIGNBELOW,ALIGNRIGHT,
FONT(SYMBOL,14)),
```

```
TEXT(B,"B",ALIGNABOVE,ALIGNLEFT,
FONT(SYMBOL,14)),TEXT(C,"C",ALIGNABOVE,ALIGNRIGHT,FONT
(TIMES,ROMAN,14)),TEXT(K,"K",ALIGNABOVE,ALIGNRIGHT,FON
T(SYMBOL,14)),CURVES([A,B,C,A],THICKNESS(3),LINESTYLE(
1)),CURVES([A,K]),AXESSTYLE(NORMAL),SCALING(CONSTRAINE
D),VIEW(-1.5..2.5,-1.5..3.5));
```



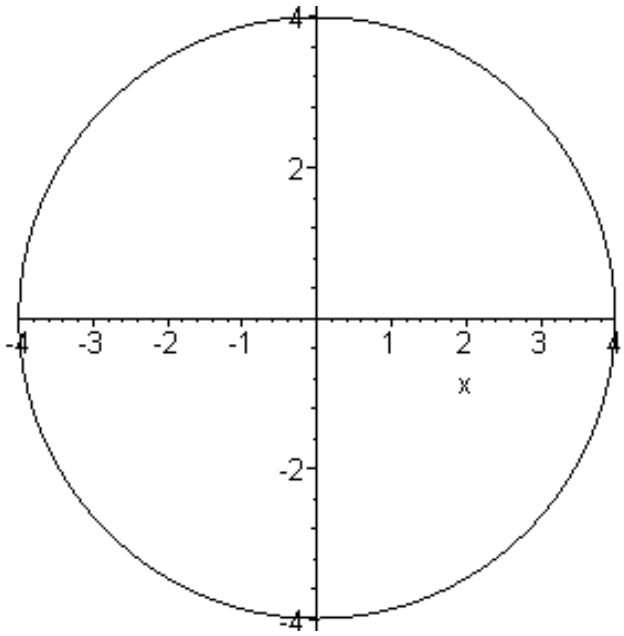
2.1.1.6 Лінії другого порядку

Важливим є вміння будувати криві другого порядку. Наведемо різні засоби Maple, призначені для подібних побудов.

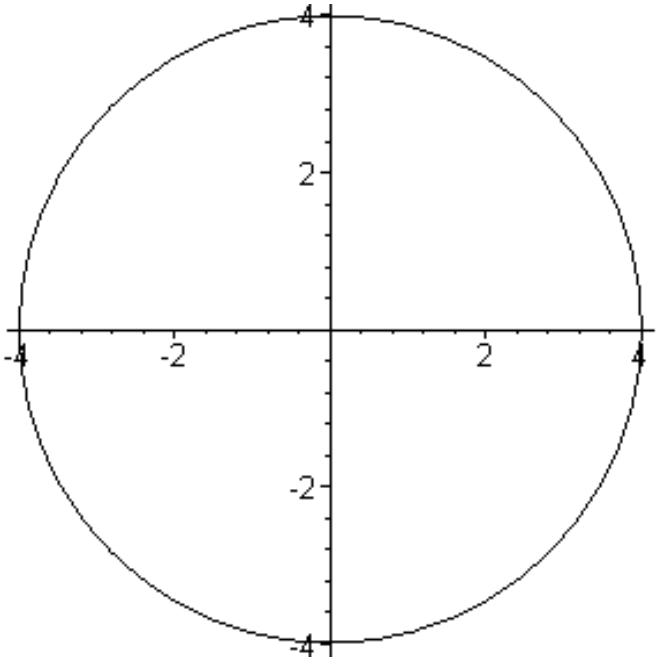
Коло

Коло радіуса r: $x^2 + y^2 = r^2$

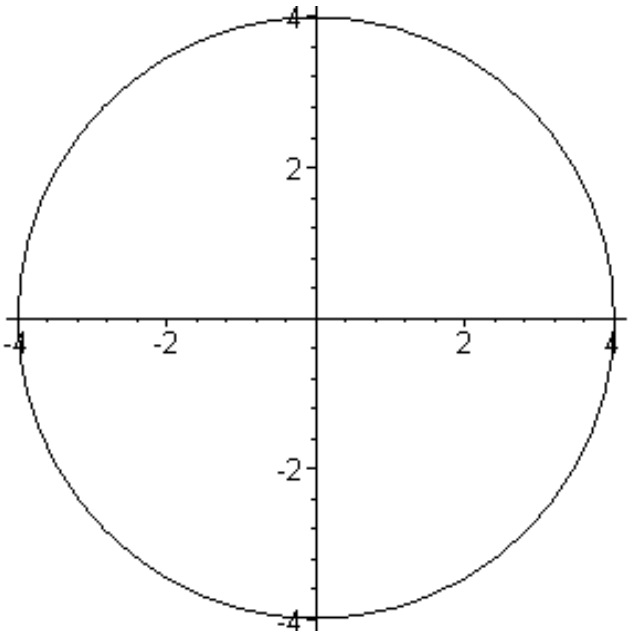
```
> r:=4:
>plot([sqrt(r^2-x^2),-sqrt(r^2-x^2)],x=-
r..r,colour=[black],scaling=CONSTRAINED);
```




```
>plot([r*sin(t),r*cos(t),t=-
Pi..Pi],scaling=CONSTRAINED);
```



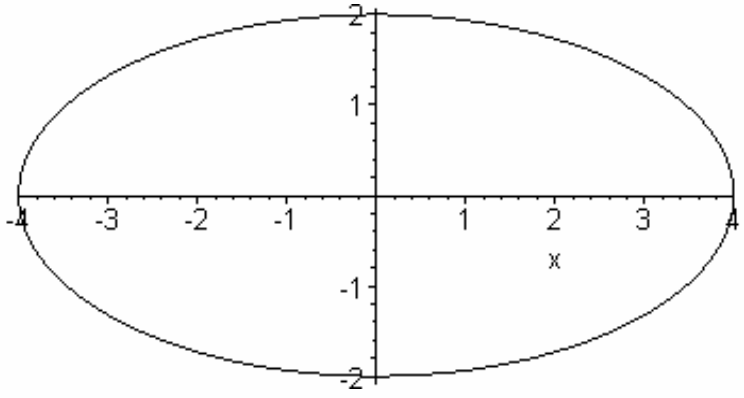

```
> plot([r,t,t=-
Pi..Pi],coords=polar,scaling=CONSTRAINED);
```



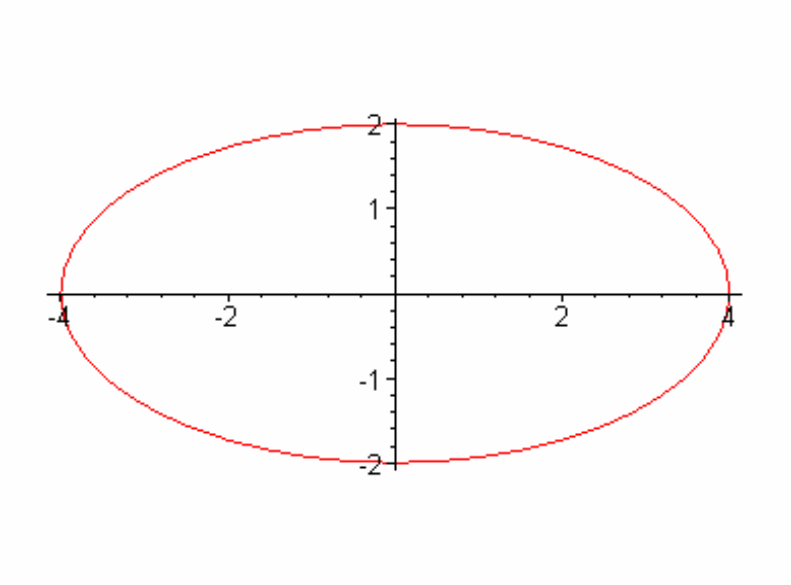
Еліпс

Рівняння еліпса: $\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$

```
> a:=4:b:=2:
>plot([b*sqrt((1-(x/a)^2)),-b*sqrt((1-(x/a)^2))],x=-
a..a,colour=[black],scaling=CONSTRAINED);
```



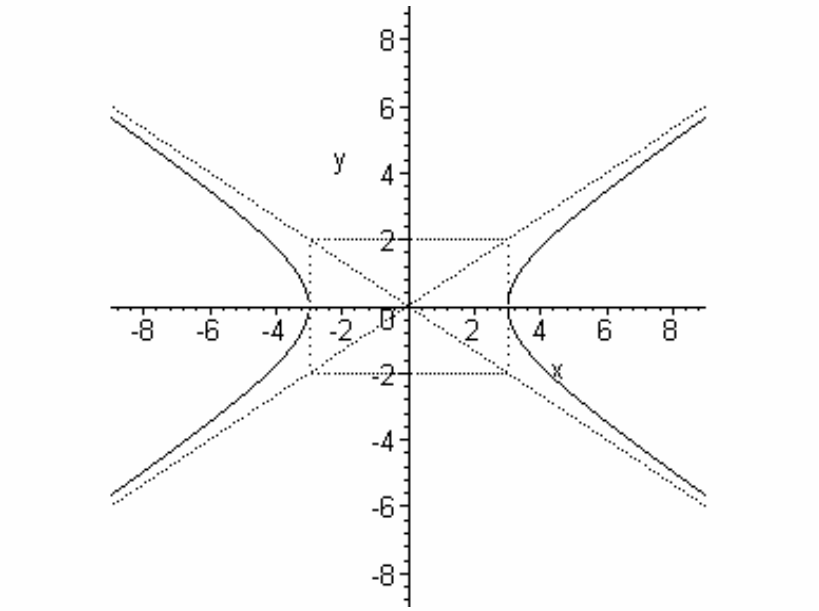

```
>plot([a*sin(t),b*cos(t),t=-
Pi..Pi],scaling=CONSTRAINED);
```



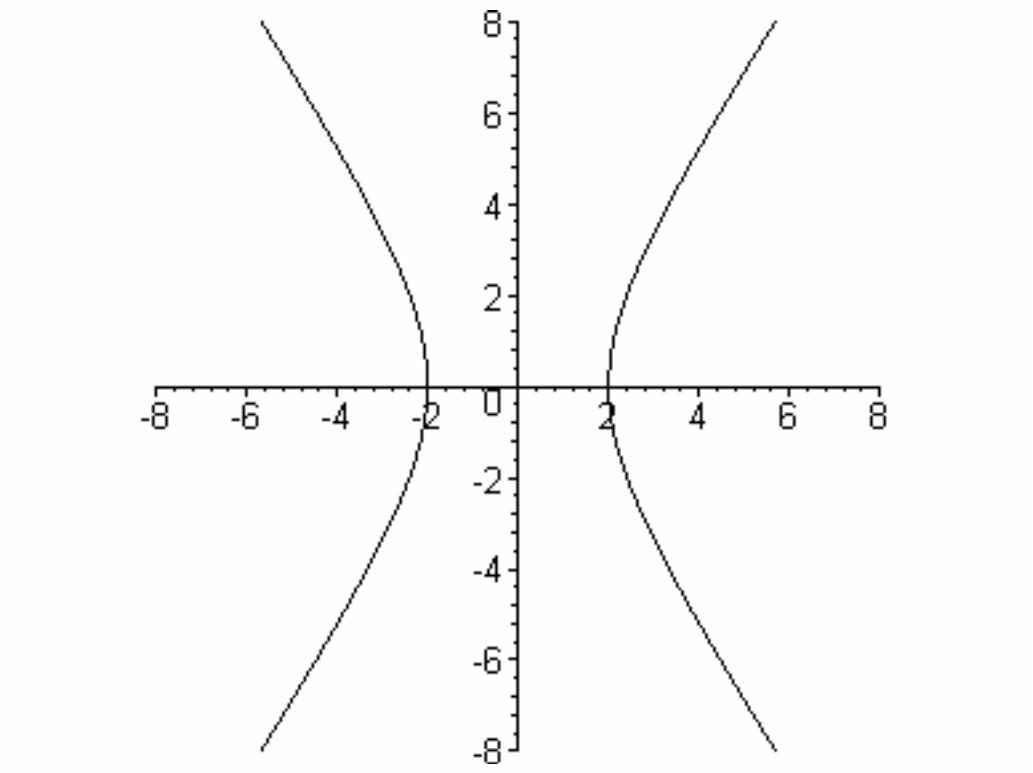
Гіпербола

Рівняння гіперболи: $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$

```
> a:=3:b:=2:
>g20:=plot([b*sqrt((-1+(x/a)^2)),-b*sqrt((-
1+(x/a)^2))],x=-3*a..3*a,colour=[black],,t=-b..b],b,-
b],x=-a..a,y=-3*a..3*a,colour=[black
scaling=CONSTRAINED):
g30:=plot([[a,t,t=-b..b],[-a,t],linestyle=2):
g40:=plot([b*x/a,-b*x/a],x=-3*a..3*a,y=-
3*a..3*a,linestyle=2,colour=[black]):
plots[display]({g20,g30,g40});
```



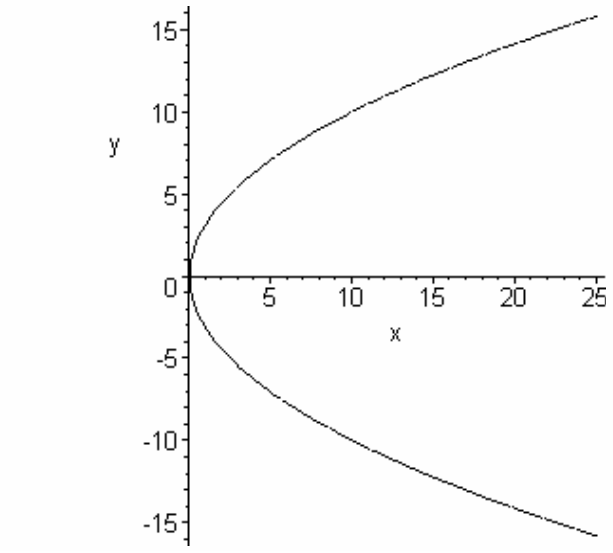
```
> plot([[b*cosh(t),a*sinh(t),  t=-Pi..Pi],[-b*cosh(t),-
a*sinh(t),  t=-
Pi..Pi]],colour=[black],scaling=CONSTRAINED,view=[-8..8, -
8..8]);
```



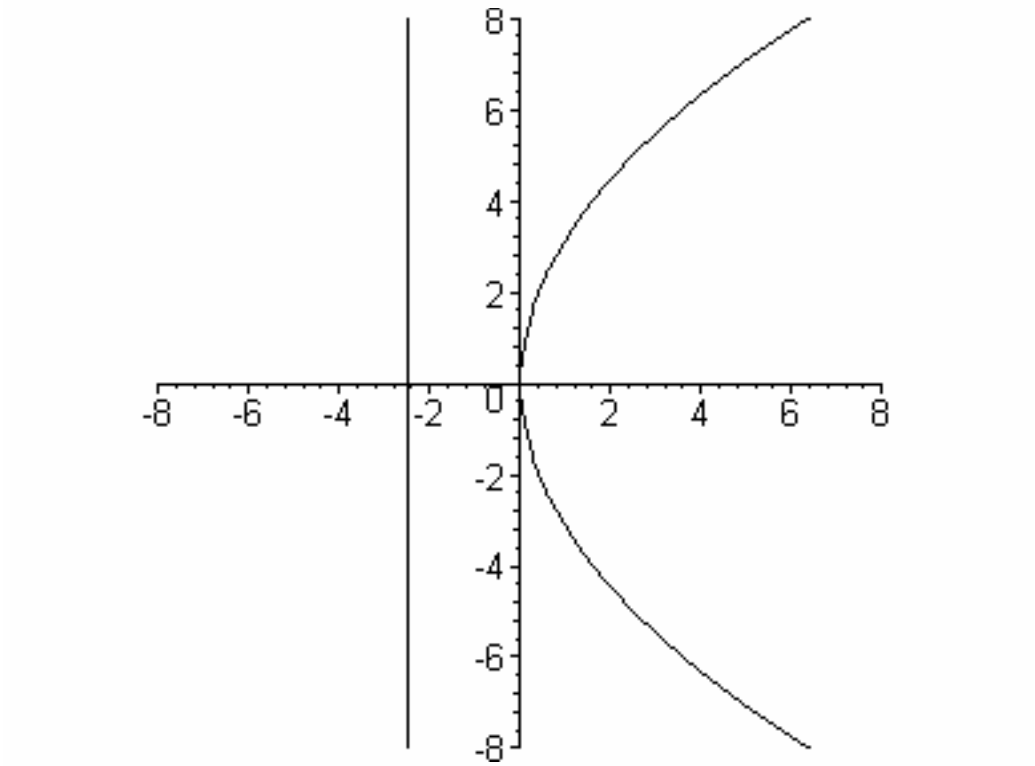
Парабола

Рівняння параболи: $y^2 = 2\,p\,x$

```
> p:=5:
>plots[implicitplot](y^2=2*p*x,x=-2*p..5*p,y=-
4*p..4*p,scaling=CONSTRAINED,colour=black);
```



```
>plot([[-p/2,t,t=-8..8],[x,sqrt(2*p*x),x=-
2*p..4*p],[x,-sqrt(2*p*x),x=-
2*p..4*p]],colour=[black],scaling=CONSTRAINED,view=[-
8..8,-8..8]);
```



2.1.1.7 Приклади розв’язання задач аналітичної геометрії в середовищі Maple

Задача 1. Рівняння однієї із сторін квадрата $x+3y-5=0$. Скласти рівняння решти трьох сторін квадрата, якщо точка з координатами (-1,0) є точкою перетину його діагоналей.

Розв'язання.

Подключаємо пакет linalg

```
> restart:with(linalg):
Warning, the protected names norm and trace have been
redefined and unprotected
```

Задання відомої сторони квадрата, перпендикулярного їй вектора і точки перетину діагоналей квадрата

(center_point):

```
> lineAB:=x+3*y-5=0;
norm_vectorAB:=vector([coeff(lhs(lineAB),x),coeff(lhs(
lineAB),y)]);
```

center_point:=vector(2,[-1,0]);

$$lineAB := x + 3\,y - 5 = 0$$

$$norm\_vectorAB := [\,1,\,3\,]$$

$$center\_point := [\,-1,\,0\,]$$

Визначивши координати двох вершин квадрата, що лежать на одній з діагоналей, можна буде записати рівняння інших сторін квадрата як рівняння прямих, що проходять через задану точку перпендикулярно (паралельно) даній прямій. Координати однієї з вершин А квадрата можна визначити як координати точки перетину двох прямих: заданої сторони квадрата і однієї з діагоналей квадрата. Рівняння діагоналі квадрата запишемо як рівняння прямої, що проходить через задану точку (-1,0) під кутом 45 градусів до заданої прямої lineAB.

> **norm_vectorAC:=vector([a,c]);#** Нормальний вектор шуканої діагоналі
point_line:=vector(2,[x,y]);# Радіус-вектор поточної точки прямої як вектор з координатами [*x*,*y*]
line_diagAC:=dotprod((point_line-center_point),norm_vectorAC)=0;# Рівняння діагоналі з невідомими *a* і *c*

$$norm\_vectorAC := [\,a,\,c\,]$$

$$point\_line := [\,x,\,y\,]$$

$$line\_diagAC := (x+1)\,\overline{a} + y\,\overline{c} = 0$$

>**Angle_45:=angle(norm_vectorAB,norm_vectorAC);**
Визначення кута між стороною і діагоналлю квадрата

$$Angle\_45 := \arccos\left(\frac{1}{10}\,\frac{(a+3\,c)\,\sqrt{10}}{\sqrt{a^2+c^2}}\right)$$

Визначення координат вершини квадрата

Як відомо кут між стороною і діагоналлю квадрата дорівнює 45 градусів. Використаємо цю умову для визначення чисельних значень невідомих *a* і *c*. Дана умова дозволить нам знайти координати нормального вектора шуканої діагоналі *a* і *c* з точністю до довжини даного вектора.

> **a:='a':eqution1:=Angle_45-Pi/4=0;#** Формування рівняння
sols1:=solve(eqution1,a);# Розв'язання рівняння відносно невідомого *a*

a:=simplify(sols1[1]);# Привласнення невідомому *a* знайденого значення

$$eqution1 := \arccos\left(\frac{1}{10}\frac{(a+3\,c)\sqrt{10}}{\sqrt{a^2+c^2}}\right)-\frac{1}{4}\pi = 0$$

$$sols1 := 2\,c, -\frac{1}{2}c$$

$$a := 2\,c$$

> **line_diagAC;#** Проглядання рівняння діагоналі з урахуванням знайденого значення *a*

$$2\,(x+1)\,\bar{c}+y\,\bar{c}=0$$

> **eqn_xy:={lineAB,line_diagAC};#** Формування системи рівнянь **sols2:=solve(eqn_xy,{x,y});#** Розв'язання системи рівнянь для визначення координат точки перетину відомої сторони квадрата з його діагоналлю

$$eqn\_xy := \{x+3\,y-5=0,\ 2\,(x+1)\,\bar{c}+y\,\bar{c}=0\}$$

$$sols2 := \{x=\frac{-11}{5}, y=\frac{12}{5}\}$$

Примітка до визначення координат вершини квадрата

Можливо, визначення вектора з точністю до його довжини може викликати додаткові утруднення в розумінні ходу розв'язання задачі. Тому тут приведемо хід розв'язання з визначенням чисельних значень координат нормального вектора діагоналі квадрата

> **a:='a':c:='c':x:='x':y:='y':**
eqution1:=Angle_45-Pi/4=0;# Формування рівнянь
eqution2:=sqrt(a^2+c^2)=1;
eqns:={eqution1,eqution2,a>=0};# Задання нерівності *a*>=0 не є необхідним і використане тільки для отримання одного і того ж з двох можливих розв'язків - виділення конкретної діагоналі з двох діагоналей квадрата.
sols1:=solve(eqns,{a,c});# Розв'язання рівняння відносно невідомих *a* і *b*

$$eqution1 := \arccos\left(\frac{1}{10}\frac{(a+3\,c)\sqrt{10}}{\sqrt{a^2+c^2}}\right)-\frac{1}{4}\pi = 0$$

$$eqution2 := \sqrt{a^2+c^2} = 1$$

$$eqns := \{\arccos\left(\frac{1}{10}\frac{(a+3\,c)\sqrt{10}}{\sqrt{a^2+c^2}}\right)-\frac{1}{4}\pi = 0, \sqrt{a^2+c^2} = 1, 0 \le a\}$$

$$sols1 := \{c = \frac{1}{10}\sqrt{10}\sqrt{2}, a = \frac{1}{5}\sqrt{10}\sqrt{2}\}$$

```
>a:=5*simplify(subs(sols1,a));
c:=5*simplify(subs(sols1, c ));
```

$$a := 2\sqrt{5}$$

$$c := \sqrt{5}$$

> **line_diagAC;#** Проглядання рівняння діагоналі з урахуванням знайдених значень *a* і *c*

$$2\,(x+1)\sqrt{5}+y\sqrt{5} = 0$$

> **eqn_xy:={lineAB,line_diagAC};#** Формування системи рівнянь
sols2:=solve(eqn_xy,{x,y});# Визначення координат точки перетину відомої сторони квадрата з його діагоналлю

$$eqn\_xy := \{x+3\,y-5 = 0, 2\,(x+1)\sqrt{5}+y\sqrt{5} = 0\}$$

$$sols2 := \{x = \frac{-11}{5}, y = \frac{12}{5}\}$$

Задамо обчислені координати вершини А квадрата у вигляді вектора

> **vectorA:=vector([eval(x,sols2),eval(y,sols2)]);**

$$vectorA := \left[\frac{-11}{5}, \frac{12}{5}\right]$$

Далі можна записати рівняння сторони AD квадрата, що проходить через вершину А перпендикулярно заданій стороні АВ. Для цього нам необхідно мати координати вектора, перпендикулярного вектору norm_vectorAB. Оскільки ми вже не вперше зустрічаємося із задачею

визначення вектора, розташованого під даним кутом до заданого, то зручніше реалізувати дану задачу у вигляді окремої процедури. Нижче приведена процедура визначення координат вектора, розташованого під даним кутом *alpha* до заданого вектора *a* .

```
> VectAng := proc (a::vector, alpha)
local n,X,A:
n:=linalg['vectdim'](a):
X:=vector(n,[seq(x[k],k=1..n)]):
A:=matrix(n,n,0):
A[1,1]:=cos(alpha):A[1,2]:=sin(alpha):
A[2,1]:=-A[1,2]:A[2,2]:=A[1,1]:
X:=evalm(a&*A);
end proc:
```

Помітимо, що в процедурі немає обмежень на кількість координат вектора. Проте коректно процедура працює лише з двовимірним вектором.

Тепер ми можемо визначити нормальний вектор прямої AD, а потім і записати її рівняння

```
> norm_vectorAD:=VectAng(norm_vectorAB,Pi/2);
lineAD:=-dotprod((point_line-
vectorA),norm_vectorAD)=0;
```

$$norm\_vectorAD := [-3, 1]$$

$$lineAD := 3x + 9 - y = 0$$

Для побудови рівнянь решти двох сторін BC і CD необхідно знайти координати вершини C квадрата. Скористаємося відомим співвідношенням для визначення координат точки M(), що ділить відрізок M1_M2 у відношенні lambda=M1_M/M_M2: *M=(M1+lambda*M2)/(1+lambda)*. Причому знак lambda додатний у разі однакового напряму векторів M1_M, M_M2 (точка M знаходиться в середині відрізка M1_M2) і від'ємний в протилежному випадку (точка M знаходиться за межами відрізка M1_M2). Точка перетину діагоналей ділить діагональ навпіл, тому

```
> lambda:=-2:
vectorC:=evalm((vectorA+lambda*center_point)/(1+lambda
));
```

$$vectorC := \left[\frac{1}{5}, \frac{-12}{5}\right]$$

Нормальні вектори прямих BC і CD рівні нормальним векторам

прямих AD і AB, тому

```
>lineBC:=-dotprod((point_line-
vectorC),norm_vectorAD)=0;
lineCD:=dotprod((point_line-vectorC),norm_vectorAB)=0;
```

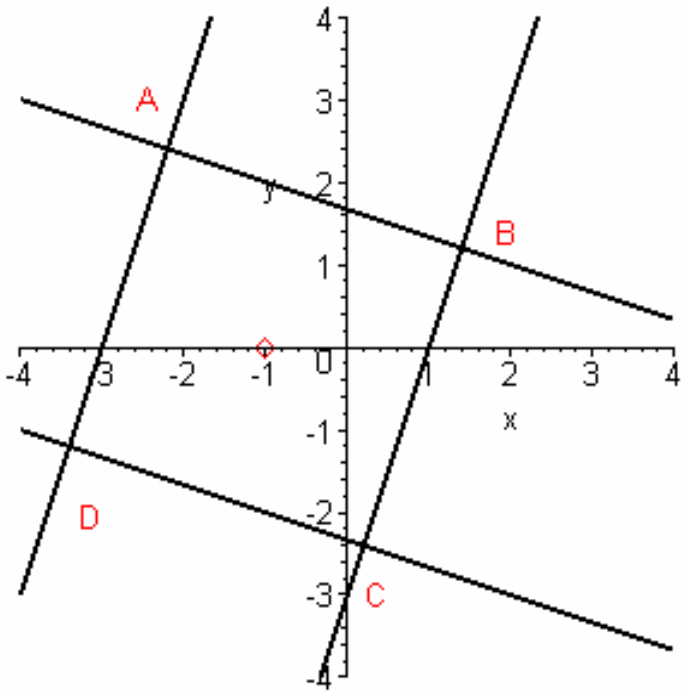
$$lineBC := 3\,x - 3 - y = 0$$

$$lineCD := x + 7 + 3\,y = 0$$

Задача вирішена. Для перевірки побудованих рівнянь сторін квадрата можна зобразити знайдені прямі лінії на графіку

```
> with(plots):
g1:=plot({rhs(isolate(lineAB,y)),
 rhs(isolate(lineBC,y)),
 rhs(isolate(lineCD,y)),
 rhs(isolate(lineAD,y))},x=-4..4,y=-
4..4,color=black,scaling=CONSTRAINED, thickness=2):
g2:=plot([[center_point[1],center_point[2]]],x=-
4..4,style=POINT,color=red,symbolsize=18):
g3:=textplot([[-2.58,3.01,"A"],[1.81,1.4,"B"],[0.22,-
3.0,"C"],[-3.28,-2.05,"D"]],color=red,align={RIGHT}):
display({g1,g2,g3},axes=NORMAL);
```

```
Warning, the name changecoords has been redefined
```



2.2 Аналітична геометрія в просторі

Розв’язання багатьох задач аналітичної геометрії в просторі з точки зору застосування можливостей Maple аналогічно розв’язанню задач аналітичної геометрії на площині.

Наведемо розв’язання двох задач.

Задача 2.2.1

Скласти рівняння площини, що проходить через три точки $M_1(x_1, y_1, z_1), M_2(x_2, y_2, z_2), M_3(x_3, y_3, z_3)$.

Вихідні дані: $M_1(1, 2, 3), M_2(-1, 3, 2), M_3(0, -3, 4)$

Розв'язання

Процедура *det_matrixS* буде потрібна далі.

```
>det_matrixS:=proc(a::matrix,j::integer,RowOrCol::stri
ng)
local y,i,m,n,detM,c;
m:=`linalg/rowdim`(a);
n:=`linalg/coldim`(a);
if m<>n then
 error("Матриця %1 повинна бути квадратною",a);
end if;
if (j<1) then
 error "Число j повинне бути натуральним числом, що не
перевищує m"
end if;
if m=1 then
 return(a[1,1]);
end if;
detM:=0;
if RowOrCol="R" then
 y:=sum('(-
1)^(j+i)*a[j,i]*det*linalg['minor'](a,j,i)',
'i'=1..m):
else
 y:=sum('(-
1)^(j+i)*a[i,j]*det*linalg['minor'](a,i,j)',
'i'=1..m):
end if
end proc:
```

Вихідні дані введемо у вигляді векторів:

```
>VectorM[1]:=[1,2,3];
VectorM[2]:=[-1,3,2];VectorM[3]:=[0, -3, 4];
#VectorM[1]:=[1,2,3];VectorM[2]:=[-1,-2,-3];
VectorM[3]:=[2, 4, 6];
```

$$VectorM_1 := [\,1, 2, 3\,]$$

$$VectorM_2 := [\,-1, 3, 2\,]$$

$$VectorM_3 := [\,0, -3, 4\,]$$

Перевіримо - чи не лежать дані точки на одній прямій. Для цього утворимо два вектори $\overrightarrow{M_1M_2}$, $\overrightarrow{M_1M_3}$ і перевіримо їх на колінеарність. Для цього знайдемо векторний добуток двох векторів

```
> M[12]:=VectorM[2]-VectorM[1]:
M[23]:=VectorM[3]-VectorM[2]:str0:=`не `:
str1:=`дорівнює нулю, отже вектори`:
str2:=` колінеарні`:str3:=` не колінеарні`:
matrix(3,3,[[i,j,k],M[12],M[23]]):
det1:=linalg[det](%):
'M[12]'=M[12],'M[23]'=M[23],det*%%=det1;
`Векторний добуток двох векторів`;
`if`(det1=0,cat(str1,str2),cat(str0,str1,str3));
```

$$M_{12} = [\,-2, 1, -1\,],\ M_{23} = [\,1, -6, 2\,],\ det\begin{bmatrix} i & j & k \\ -2 & 1 & -1 \\ 1 & -6 & 2 \end{bmatrix} = -4\,i + 3\,j + 11\,k$$

Векторний добуток двох векторів

не дорівнює нулю, отже вектори не колінеарні

Це означає що задані три точки *M1*, *M2*, *M3*

```
> str4:=`належать одній прямій.`:
`if`(det1=0,str4,cat(str0,str4));
```

не належать одній прямій.

Запишемо рівняння площини у вигляді рівності нулю мішаного добутку трьох векторів:

```
> X:=[x,y,z]:
MatrixPlain:=matrix([X-VectorM[1],VectorM[2]-
VectorM[1],VectorM[3]-
VectorM[1]]):seq(sort(MatrixPlain[1,k]),k=1..3):
det*evalm(MatrixPlain)=0;
EqPlain:=linalg[det](MatrixPlain)=0:
```

$$det\begin{bmatrix} x-1 & y-2 & z-3 \\ -2 & 1 & -1 \\ -1 & -5 & 1 \end{bmatrix} = 0$$

За допомогою формули розкладання визначника за елементами першого рядка, матимемо

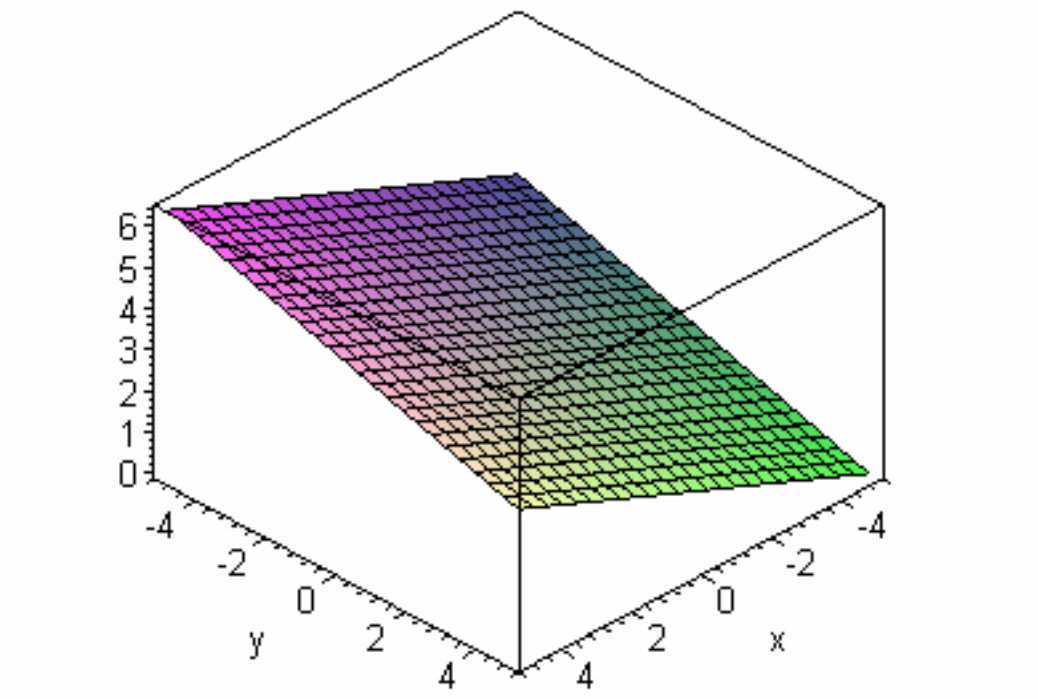
```
> det_matrixS(MatrixPlain,1,"R")=0;
EqPlain:=sort(EqPlain,[x,y,z]):
if lhs(EqPlain)=0 then
 print(`Задані точки лежать на одній прямій, отже через них можна провести безліч різних площин.`)
else
 print(`Рівняння площини`);
 EqPlain
end if;
```

$$(x-1)\,det\begin{bmatrix} 1 & -1 \\ -5 & 1 \end{bmatrix} - (y-2)\,det\begin{bmatrix} -2 & -1 \\ -1 & 1 \end{bmatrix} + (z-3)\,det\begin{bmatrix} -2 & 1 \\ -1 & -5 \end{bmatrix} = 0$$

Рівняння площини

$$-4x + 3y + 11z - 35 = 0$$

```
> plot3d(solve(EqPlain,z),x=-5..5,y=-5..5);
```



Задача 2.2.2

Написати рівняння площини, яка паралельна площині $5x-14y+2z+36=0$ і віддалена від неї на задану відстань d.

Розв'язання 1. Задамо задану площину таким чином. Нормальний вектор даної площини

```
> restart:with(linalg):
NormVector:=vector([5,-14,2]);
point_plain:=vector(3,[x,y,z]);#          Радіус-вектор
поточної точки площини як вектор з координатами
[x,y,z]
```

```
Warning, the protected names norm and trace have been
redefined and unprotected
```

$$NormVector := [5, -14, 2]$$

$$point\_plain := [x, y, z]$$

```
> plain:=dotprod(point_plain,NormVector)+36=0;#Рівняння
заданої площини
```

$$plain := 5x - 14y + 2z + 36 = 0$$

Шукану площину задамо у вигляді загального рівняння площини з невідомими коефіцієнтами

```
> s:=A*x+B*y+C*z+D=0;
```

$$s := Ax + By + Cz + D = 0$$

Оскільки за умовою шукана площина паралельна заданій, то нормальні вектори цих площин рівні, тому

```
> A:=op(1,op(1,lhs(plain)));
B:=op(1,op(2,lhs(plain)));
C:=op(1,op(3,lhs(plain)));
```

$$A := 5$$

$$B := -14$$

$$C := 2$$

Помітимо, що запропонований варіант визначення значень A, B, C порівняно з простим заданням (A=5...) має перевагу спільності для різних варіантів. Ця перевага виявляється, якщо в умові задачі змінити значення координат нормального вектора заданої площини. При простому заданні

значень A, B, C їх потрібно було б перевизначати наново "вручну". В нашому випадку це відбувається програмно.
Рівняння шуканої площини набуває вигляду

```
> s;
```

$$5\,x - 14\,y + 2\,z + \mathrm{D} = 0$$

Нам залишилося визначити невідому величину D. Для цього необхідно використовувати задану відстань **d** між площинами. Подальший алгоритм розв'язання може бути таким. Вибираємо довільну точку на заданій площині, наприклад M(0, 0, zм), проводимо через цю точку пряму, перпендикулярну заданій площині. На отриманій прямій знаходимо точку N, віддалену від точки M на відстань **d**. І будуємо рівняння шуканої площини як площини, що проходить через задану точку в заданому напрямі.

```
>pointM:=([0,0,-
op(4,lhs(plain))/op(1,op(3,lhs(plain)))]);
point_line:=vector([X,Y,Z]);# Радіус-вектор поточної точки лінії як вектор з координатами [X,Y,Z]
line_ort:=evalm(point_line)=pointM+evalm(t*NormVector)
;# Рівняння лінії, що проходить перпендикулярно заданій площині 5x-14y+2z+36=0 через точку pointM(0,0, -18), яка належить цій площині
sols1:=solve(sum((t*NormVector[k])^2,k=1..3)=d^2,t);#
Пошук значення параметра t, якому відповідає точка на лінії line_ort
```

$$pointM := [\,0,\,0,\,-18\,]$$

$$point\_line := [\,X,\,Y,\,Z\,]$$

$$line\_ort := [\,X,\,Y,\,Z\,] = [\,0,\,0,\,-18\,] + [\,5\,t,\,-14\,t,\,2\,t\,]$$

$$sols1 := -\frac{1}{15}\,d,\ \frac{1}{15}\,d$$

Як видно, існує два розв'язки. Це означає, що можна побудувати дві площини, які паралельні заданій і розташовані на заданій відстані. Визначимо шукані точки N1, N2 і рівняння відповідних площин

```
>point_N1:=evalm(pointM+sols1[1]*NormVector);point_N2:
=evalm(pointM+sols1[2]*NormVector);
print(`Перше рівняння шуканої площини`);
s1:=sort(dotprod((point_plain-
point_N1),NormVector),[x,y,z])=0;# 1 - е рівняння
шуканої площини
```

```
print(`Друге рівняння шуканої площини`);
s2:=sort(dotprod((point_plain-
point_N2),NormVector),[x,y,z])=0;# 2 - е рівняння
шуканої площини
```

$$point\_N1 := \left[-\frac{1}{3}d, \frac{14}{15}d, -18 - \frac{2}{15}d \right]$$

$$point\_N2 := \left[\frac{1}{3}d, -\frac{14}{15}d, -18 + \frac{2}{15}d \right]$$

Перше рівняння шуканої площини

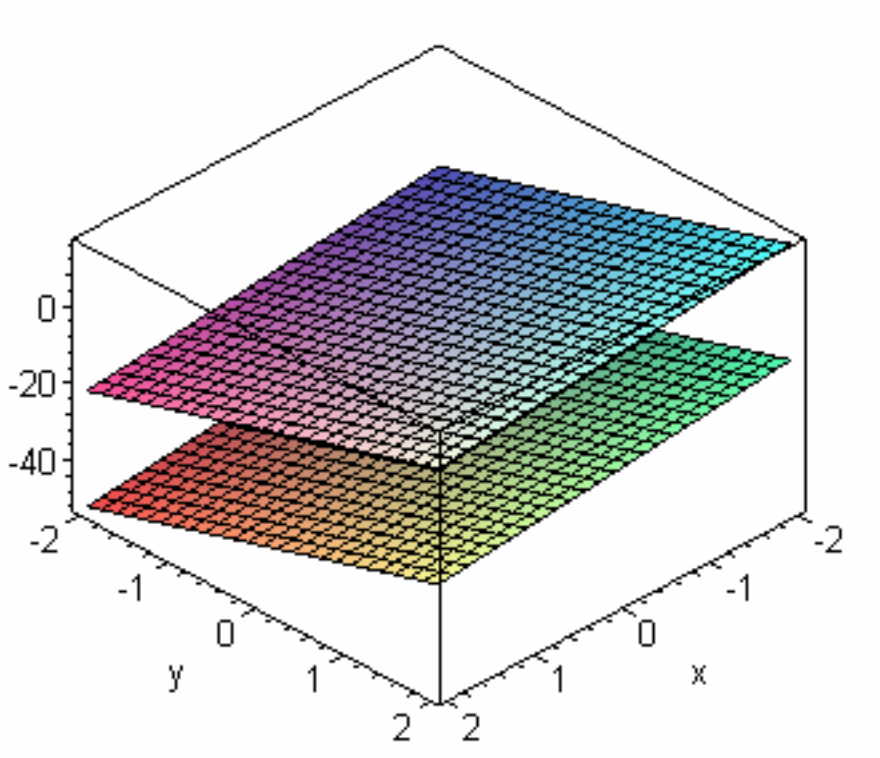
$$s1 := 5x - 14y + 2z + 15d + 36 = 0$$

Друге рівняння шуканої площини

$$s2 := 5x - 14y + 2z - 15d + 36 = 0$$

Присвоємо величині *d* значення 2 та побудуємо отримані площини.

```
>plot3d({subs(d=2,solve(s1,z)),subs(d=2,solve(s2,z))},
x=-2..2,y=-2..2);
```



2.2.3 Поверхні другого порядку

Maple має потужні засоби для побудови графіків функцій двох змінних. Надзвичайно складні графіки можна побудувати за допомогою стандартної команди plot3d(). Але в багатьох, навіть простих випадках,

потрібно вміти користуватися більш широкими засобами *Maple* для того, щоб отримати наочні поверхні.

2.2.3.1 Конус:

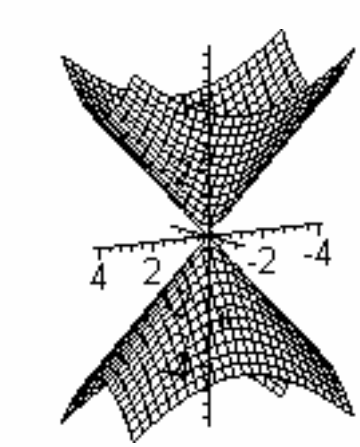
$$\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}=0.$$

Спробуємо зобразити конічну поверхню безпосереднім застосуванням стандартної команди

```
> restart:
a:=2:b:=1:c:=2:
> `КОНУС`;
x^2/(a^2)+y^2/(b^2)-z^2/(c^2) = 0;
plot3d({c*sqrt(x^2/(a^2)+y^2/(b^2)),-
c*sqrt(x^2/(a^2)+y^2/(b^2))},x=-2*a..2*a,y=-2*b..2*b,
color=white,axes=normal,orientation=[57,80],
scaling=constrained);
```

КОНУС

$$\frac{1}{4}x^2+y^2-\frac{1}{4}z^2=0$$



Як видно, простим використанням стандартної команди побудови поверхонь виходить графік, який не дає наочного уявлення про конічну поверхню.

Задану конічну поверхню

```
x^2/(a^2)+y^2/(b^2)-z^2/(c^2) = 0;
```

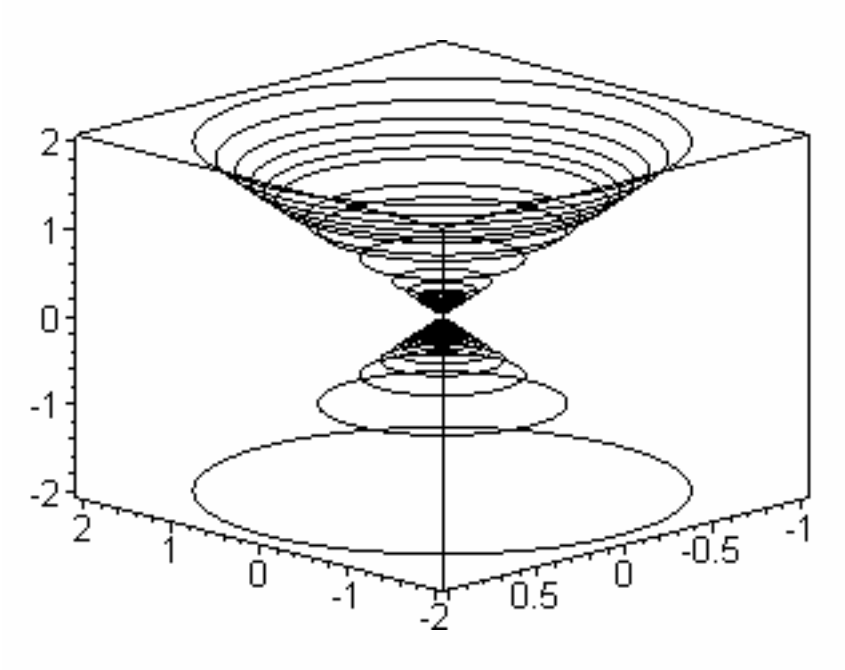
$$\frac{1}{4}x^2+y^2-\frac{1}{4}z^2=0$$

спробуємо зобразити іншим способом.

```
> xt := a * cos(t):  yt := b*sin(t):Zconst:=c:
```

Будуємо послідовність еліпсів, що є перетинами заданої конічної поверхні з різними площинами, перпендикулярними осі Z

```
>planeCurve:=
plots[spacecurve]({seq([xt/i,yt/i,c/i],i=[1,1.1,1.2,1.
3,1.4,1.5,1.8,2,3,5,7,9,10,11,14,27,30]),seq([xt/i,yt/
i,-c/i],i=1..30)},       t=0..2*Pi,       color=black,
thickness=1,     axes=boxed,     orientation=[135,70],
scaling=unconstrained):
planeCurve;
```

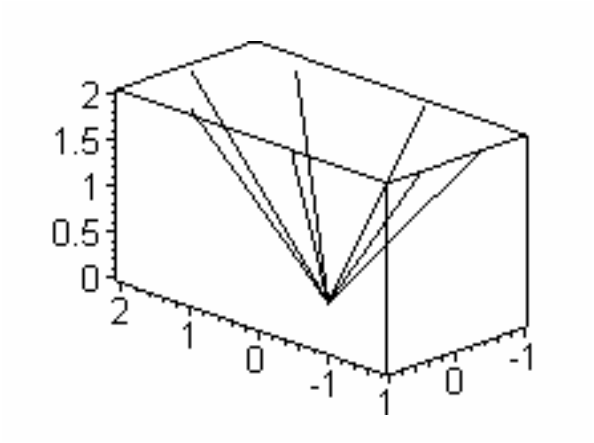



```
> numLines := 7:
for i from 1 to numLines do
 t := (i/numLines)*2*Pi;
 lines1[i]:=plottools[line]([0,0,0],          [xt,yt,c],
color=black, linestyle=1, thickness=1);
lines2[i]:=plottools[line]([0,0,0],         [xt,yt,-c],
color=black, linestyle=1, thickness=1);
end do:
```

Далі будуємо поверхню послідовністю прямих, що повернені на деякий кут навколо осі z:

```
>surfaceCurve1 := plots[display]({seq(lines1[i],
i=1..numLines)}, axes=boxed, scaling=constrained,
```

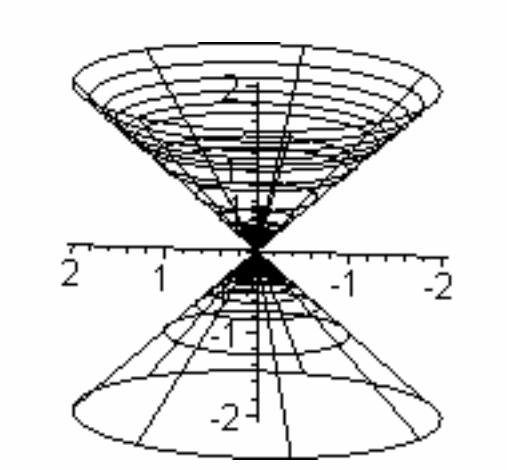
```
orientation=[135,70]):
surfaceCurve2 := plots[display]({seq(lines2[i],
i=1..numLines)}, axes=boxed, scaling=constrained,
orientation=[135,70]):
surfaceCurve1;
```




```
> `КОНУС`;
x^2/(a^2)+y^2/(b^2)-z^2/(c^2) = 0;
plots[display]({planeCurve, surfaceCurve1,surfaceCurve2},
axes=normal, scaling=constrained, orientation=[94,62]);
```

КОНУС

$$\frac{1}{4}x^2 + y^2 - \frac{1}{4}z^2 = 0$$



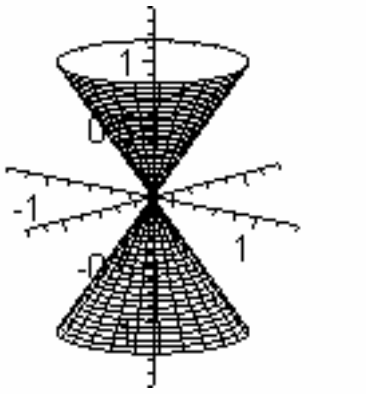
Один із альтернативних і найпростіших способів побудови конуса такий

```
> with(plottools):
ty:=1.4:
My_cone:=plottools[cone]([0,0,0],0.7,1,color=white),co
ne([0,0,0],0.7,-1,color=white):
`КОНУС`;
```

```
x^2/(a^2)+y^2/(b^2)-z^2/(c^2) = 0;
plots[display](My_cone,scaling=constrained,axes=normal
, orientation=[41,77], style=patch,view=[-ty..ty,-
ty..ty,-ty..ty]);
```

КОНУС

$$\frac{1}{4}x^2 + y^2 - \frac{1}{4}z^2 = 0$$



2.2.3.2 Еліптичний параболоїд

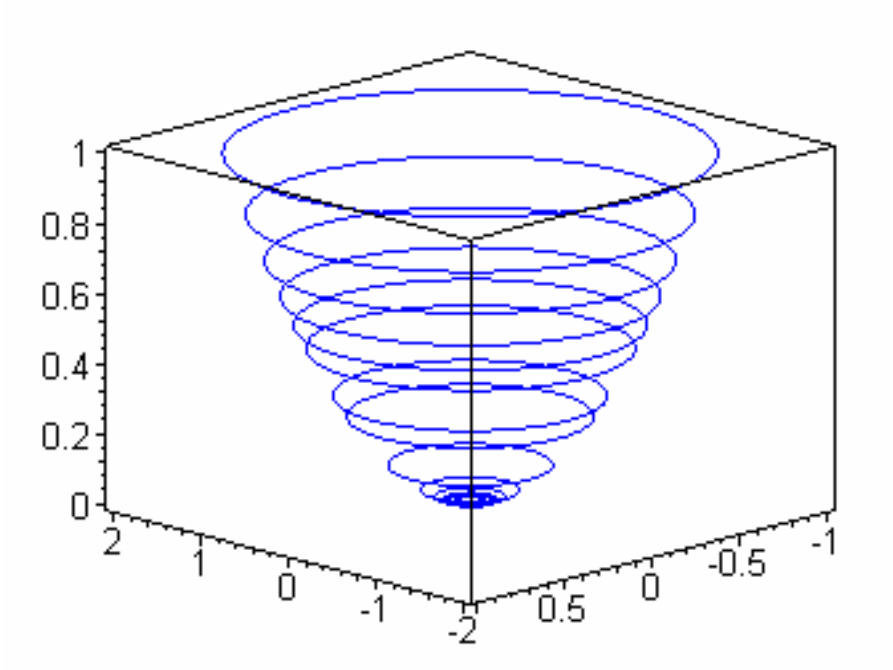
```
> restart:
x^2/(a^2)+y^2/(b^2)=2*p*z;
a:=2:b:=1:p:=1/2:
```

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 2\,p\,z$$

```
> xt := a * cos(t):  yt := b*sin(t):
```

Будуємо послідовність еліпсів, що є перетинами заданої конічної поверхні з різними площинами, перпендикулярними осі Z

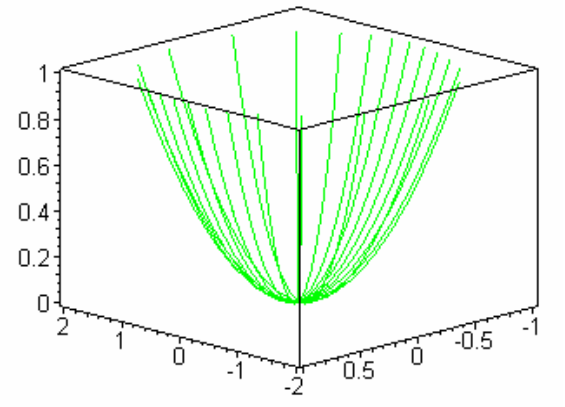
```
>planeCurve:=plots[spacecurve]({seq([xt/i,yt/i,(1/i)^2
],i=[1,1.1,1.2,1.3,1.4,1.5,1.8,2,3,5,7,9,10,11,14,27,3
0])},t=0..2*Pi,color=blue,thickness=1,axes=boxed,orien
tation=[135,70],scaling=unconstrained):
planeCurve;
```




```
>delta:=Pi/180:m:=15:
den:=sqrt(1+4*tan(delta*m*alpha)^2):
X := [seq(evalf(xt/den), alpha=-86/m..86/m)]:
Y := [seq(evalf(tan(delta*m*alpha)*xt/den), alpha=-
86/m..86/m)]:
Z := [seq(evalf(((X[i]/a)^2+(Y[i]/b)^2)/(2*p)),
i=1..nops(X))]:
Seq1:={seq( [X[i],Y[i],Z[i]], i=1..nops(X))}:
```

Далі будуємо поверхню послідовністю прямих, що повернені на деякий кут навколо осі z:

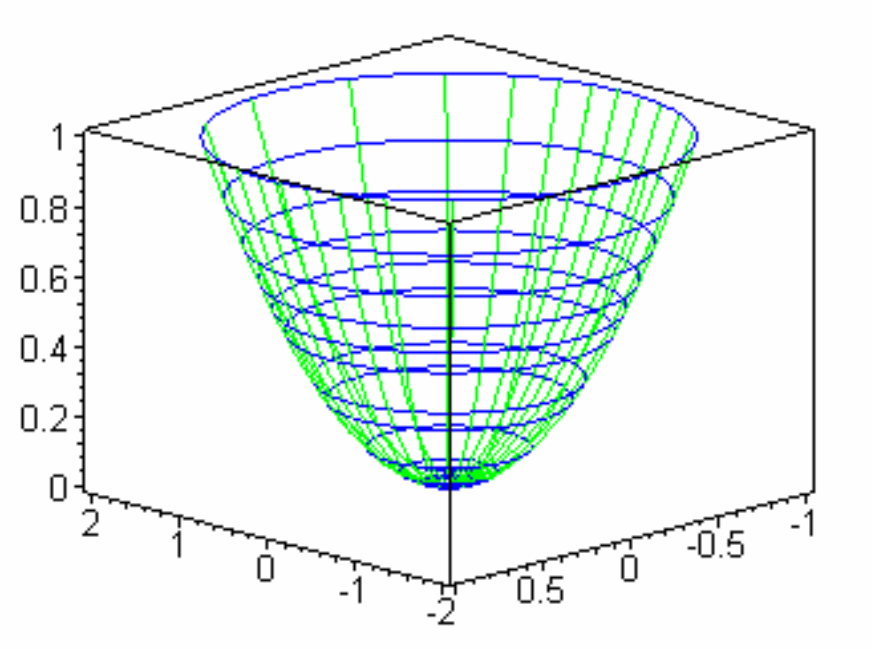
```
> surfaceCurve := plots[spacecurve](Seq1, t=0..2*Pi,
color=green,
thickness=1,orientation=[135,70],scaling=UNCONSTRAINED
,axes=boxed):
surfaceCurve;
```



```
> `Еліптичний параболоїд`;
x^2/(a^2)+y^2/(b^2)=2*p*z;
plots[display]({planeCurve,surfaceCurve},axes=boxed,
scaling=unconstrained, orientation=[135,70]);
```

Еліптичний параболоїд

$$\frac{1}{4}x^2 + y^2 = z$$



2.2.3.3 Однопорожнинний гіперболоїд

```
> restart:
x^2/(a^2)+y^2/(b^2)-z^2/(c^2) = 1;
a:=2:b:=1:c:=3:
```

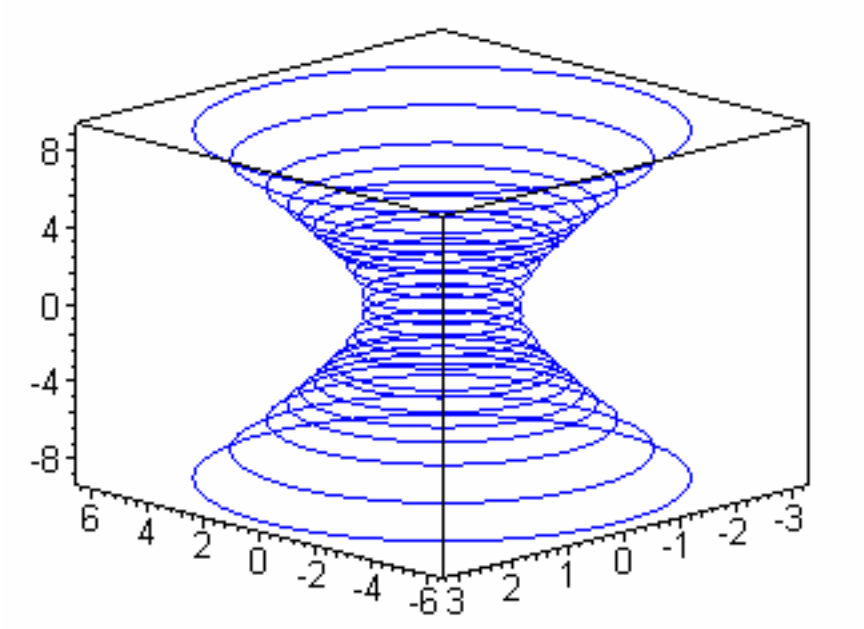
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$$

```
> xt := a * cos(t):  yt := b*sin(t):Zconst:=c:
```

Будуємо послідовність еліпсів, що є перетинами заданої конічної поверхні з різними площинами, перпендикулярними осі Z

```
>planeCurve:=plots[spacecurve]({seq([xt*sqrt(1+i^2),yt
*sqrt(1+i^2),c*i],i=[0,0.2,0.5,0.7,1,1.1,1.1,1.3,1.5,1
.7,2,2.5,3]),seq([xt*sqrt(1+i^2),yt*sqrt(1+i^2),-
c*i],i=[0,0.2,0.5,0.7,1,1.1,1.1,1.3,1.5,1.7,2,2.5,3])}
,          t=0..2*Pi,color=blue,thickness=1,axes=boxed,
```

```
orientation=[135,70], scaling=unconstrained):
planeCurve;
```




```
>
delta:=Pi/180:m:=15:den:=sqrt(1+4*tan(delta*m*alpha)^2
):
X := [seq(evalf(xt/den)*sqrt(10), alpha=-86/m..86/m)]:
Y := [seq(evalf(tan(delta*m*alpha)*xt/den)*sqrt(10),
alpha=-86/m..86/m)]:
Z := [seq(evalf(c*sqrt((X[i]/a)^2+(Y[i]/b)^2-1)),
i=1..nops(X))]:
Seq1:={seq( [X[i],Y[i],Z[i]], i=1..nops(X))}:
Seq2:={seq( [X[i],Y[i],-Z[i]], i=1..nops(X))}:
Seq1:
```

Далі будуємо поверхню послідовністю прямих, що повернені на деякий кут навколо осі z:

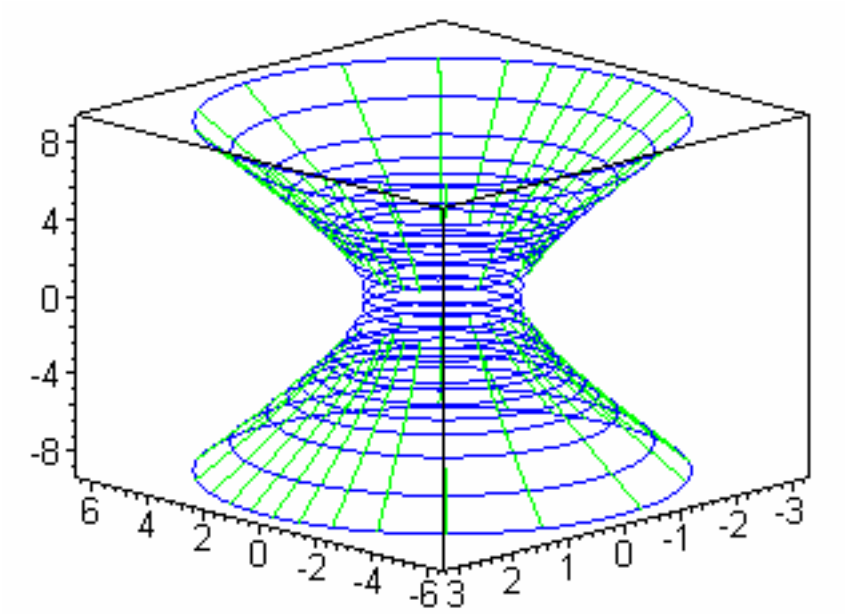
```
>surfaceCurve1 := plots[spacecurve](Seq1, t=0..2*Pi,
color=green,
thickness=1,orientation=[135,70],scaling=UNCONSTRAINED
, axes=boxed):
surfaceCurve2 := plots[spacecurve](Seq2, t=0..2*Pi,
color=green,
thickness=1,orientation=[135,70],scaling=UNCONSTRAINED
, axes=boxed):
#surfaceCurve1;surfaceCurve2;plots[display]({surfaceCu
rve1,surfaceCurve2},axes=boxed,scaling=unconstrained,
orientation=[135,70]):

> `Однопорожнинний гіперболоїд`;
```

```
x^2/(a^2)+y^2/(b^2)-z^2/(c^2) = 1;
plots[display]({planeCurve,surfaceCurve1,surfaceCurve2
},          axes=boxed,          scaling=unconstrained,
orientation=[135,70]);
```

Однопорожнинний гіперболоїд

$$\frac{1}{4}x^2 + y^2 - \frac{1}{9}z^2 = 1$$



Однопорожнинний гіперболоїд обертання.

```
x^2/('b'^2)+y^2/('b'^2)-z^2/('c'^2) = 1;
```

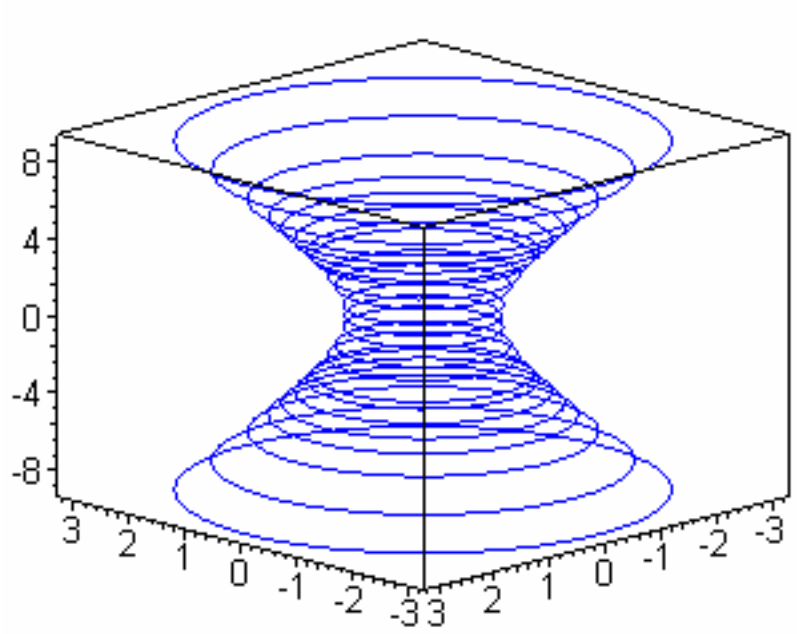
$$\frac{x^2}{b^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$$

```
> xt := b * cos(t): yt := b*sin(t):Zconst:=c:
```

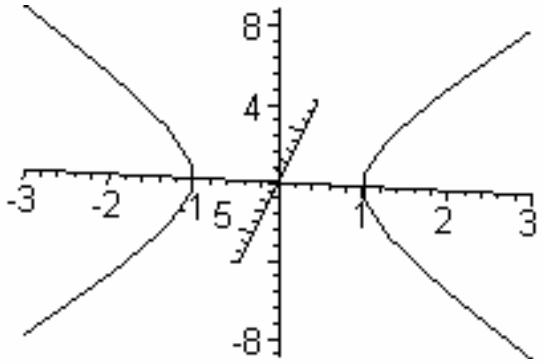
Будуємо послідовність еліпсів, що є перетинами заданої конічної поверхні з різними площинами, перпендикулярними осі Z

```
> with(plottools):
with(plots):
planeCurve                                                    :=
plots[spacecurve]({seq([xt*sqrt(1+i^2),yt*sqrt(1+i^2),
c*i],i=[0,0.2,0.5,0.7,1,1.1,1.1,1.3,1.5,1.7,2,2.5,3]),
seq([xt*sqrt(1+i^2),yt*sqrt(1+i^2),-
c*i],i=[0,0.2,0.5,0.7,1,1.1,1.1,1.3,1.5,1.7,2,2.5,3])}
,          t=0..2*Pi,color=blue,thickness=1,axes=boxed,
orientation=[135,70], scaling=unconstrained):
```

```
planeCurve;
```
```
Warning, the names arrow and changecoords have been
redefined
```

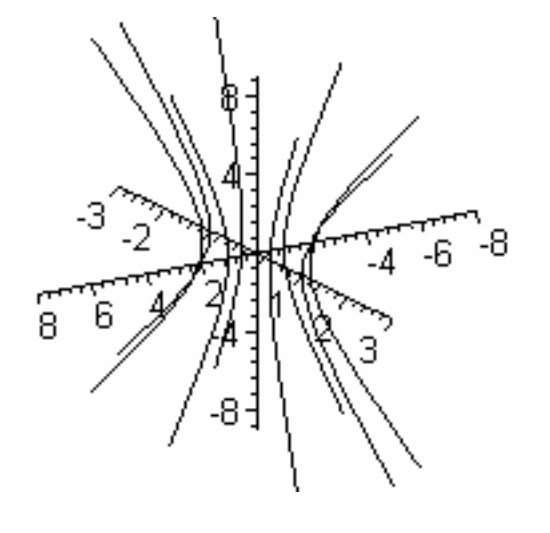



```
> xt1:0:yt1:=b*cosh(t):zt1:=c*sinh(t):
plots[spacecurve]({[0,yt1,zt1],[0,-yt1,zt1]},t=-
4*Pi..4*Pi,color=black,view=[-4*a..4*a,-3*b..3*b,-
3*c..3*c], orientation=[9,65],axes=normal);
Gipr[0]:=spacecurve({[0,yt1,zt1],[0,-yt1,zt1]},t=-
4*Pi..4*Pi,color=black,view=[-4*a..4*a,-3*b..3*b,-
3*c..3*c], orientation=[9,65],axes=normal):
```




```
> l := [[0,0,0],[0,0,1]]:
aa := Pi/4: cc := 1:
while evalf(aa-2*Pi) < 0 do
  Gipr[cc] := rotate(Gipr[0],aa,l);
  aa := aa+Pi/4;
  cc := cc+1;
```

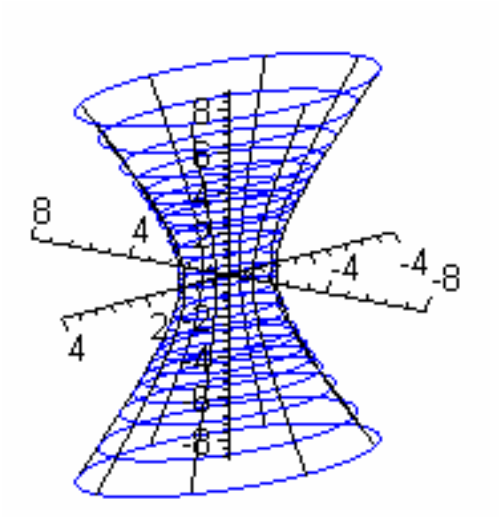
```
end do:
plots[display]([seq(Gipr[i],i=0..cc-1)]);
```




```
> `Однопорожнинний гіперболоїд обертання`;
x^2/(b^2)+y^2/(b^2)-z^2/(c^2) = 1;
plots[display]({planeCurve,    seq(Gipr[i],i=0..cc-1)},
axes=boxed,     scaling=unconstrained,view=[-4*a..4*a,-
4*b..4*b,-3*c..3*c],
orientation=[123,70],axes=normal);
```

Однопорожнинний гіперболоїд обертання

$$x^2 + y^2 - \frac{1}{9}z^2 = 1$$

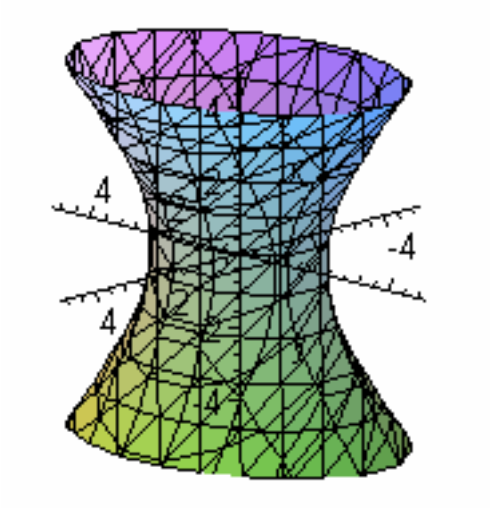


Мабуть, найпростіший спосіб побудови поверхні - за допомогою команди *smartplot3d*. Цю команду можна вибрати із контекстного меню, яке з'являється, якщо виділити рівняння поверхні (в зоні Text Output) та клацнути на виділеному правою кнопкою миші.

```
> `Однопорожнинний гіперболоїд`;
1/4*x^2+1/2*y^2-1/9*z^2 = 1;
smartplot3d[x,y,z](1/4*x^2+1/2*y^2-1/9*z^2 = 1);
```

Однопорожнинний гіперболоїд

$$\frac{1}{4}x^2+\frac{1}{2}y^2-\frac{1}{9}z^2=1$$



2.2.3.4 Гіперболічний параболоїд

```
> restart:
HP:=x^2/(a^2)-y^2/(b^2)=2*p*z:
HP;a:=3:b:=2:p:=1/20:HP;
```

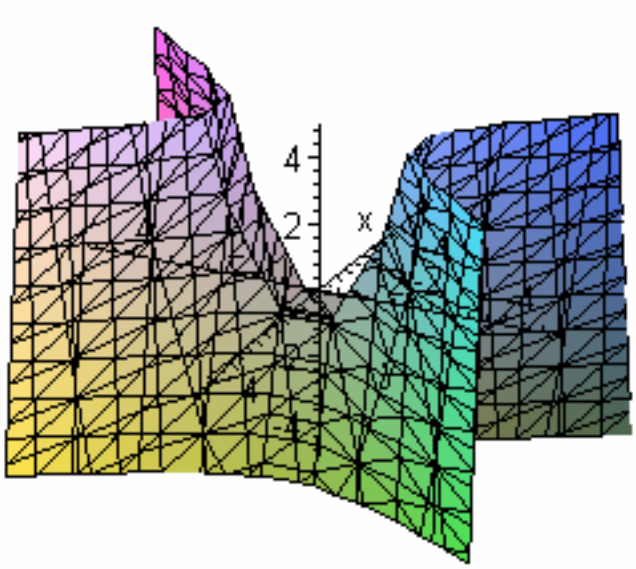
$$\frac{x^2}{a^2}-\frac{y^2}{b^2}=2\,p\,z$$

$$\frac{1}{9}x^2-\frac{1}{4}y^2=\frac{1}{10}z$$

```
> `Гіперболічний параболоїд`;
1/9*x^2-1/4*y^2 = 1/10*z;
smartplot3d[x,y,z](1/9*x^2-1/4*y^2 = 1/10*z);
with(plots):
dp:=4:
`Гіперболічний параболоїд`;
1/9*x^2-1/4*y^2 = 1/10*z;
plot3d(10/9*x^2-10/4*y^2,x=-4..4,y=-4..4,view=-4..4,
orientation=[112,57],axes=normal);
```

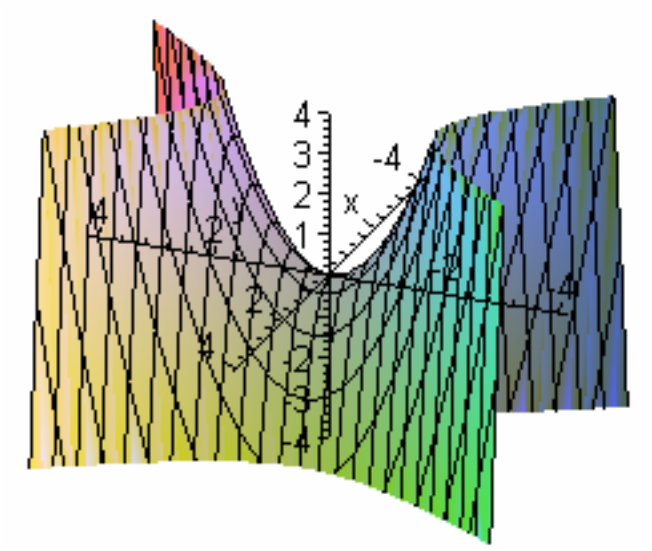
Гіперболічний параболоїд

$$\frac{1}{9}x^2 - \frac{1}{4}y^2 = \frac{1}{10}z$$

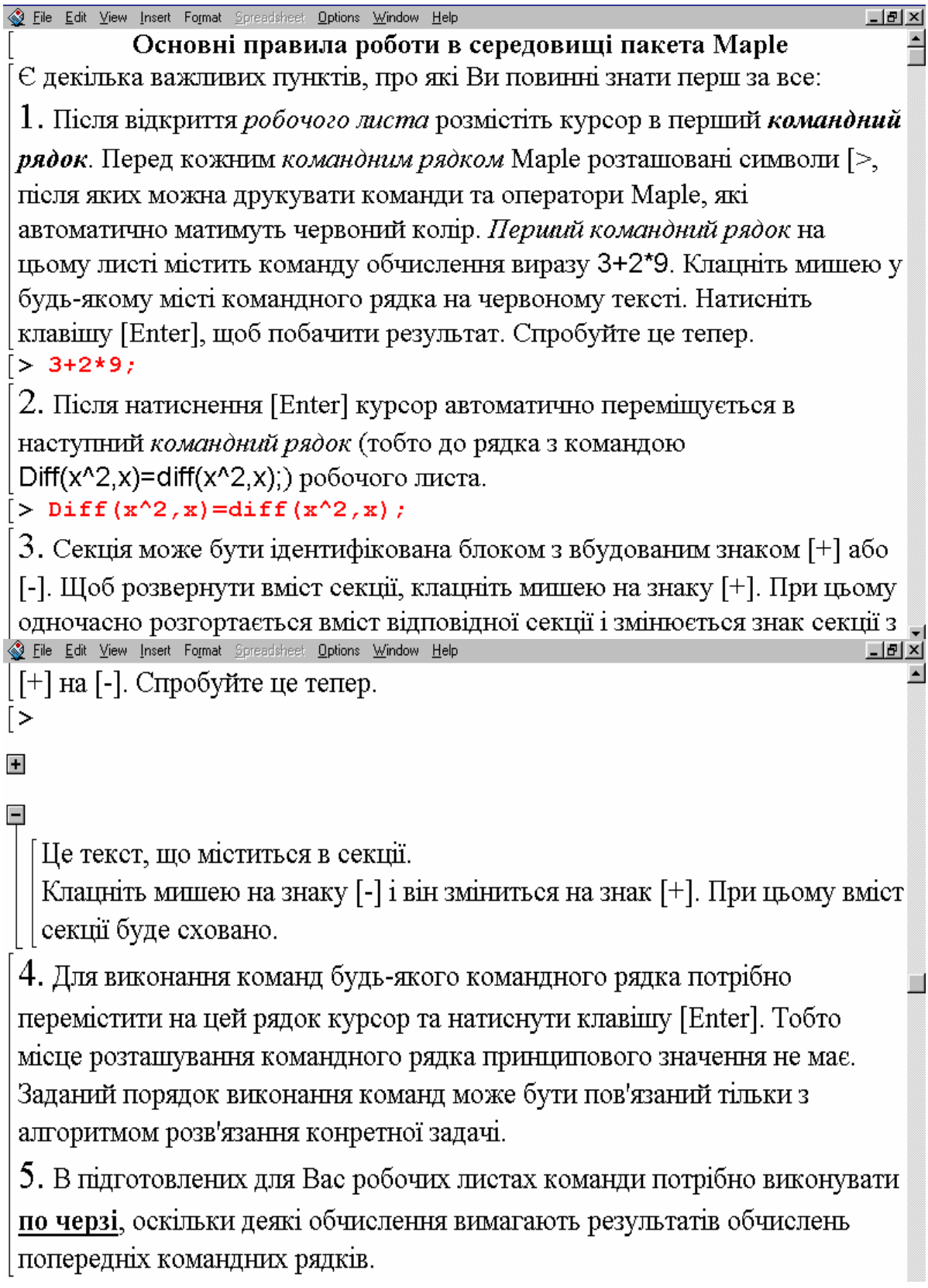


Гіперболічний параболоїд

$$\frac{1}{9}x^2 - \frac{1}{4}y^2 = \frac{1}{10}z$$



3. Приклади електронних лекцій в Maple



Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

1 Матриці

1.1 Поняття матриці

Матрицею називається прямокутна таблиця

$$A=\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \quad (1),$$

де a_{11}, a_{12}, ..., a_{mn} - числа або букви, які називаються елементами матриці.

Елементи a_{11}, a_{12}, ..., a_{1n} утворюють перший рядок матриці, елементи a_{21}, a_{22}, ..., a_{2n} - другий рядок і т.д.; елементи a_{11}, a_{21}, ..., a_{m1} утворюють перший стовпець матриці, елементи a_{12}, a_{22}, ..., a_{m2} - другий стовпець матриці і т.д.

Елемент матриці a_{ij} - елемент, що стоїть на перетині i-го рядка та j-го стовпця матриці.

Матриця (1) має m рядків та n стовпців, тобто є матрицею розмірності

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

$m \times n$.

Крім квадратних дужок застосовують і такі позначення для матриць:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \text{ або } \begin{Vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{Vmatrix}.$$

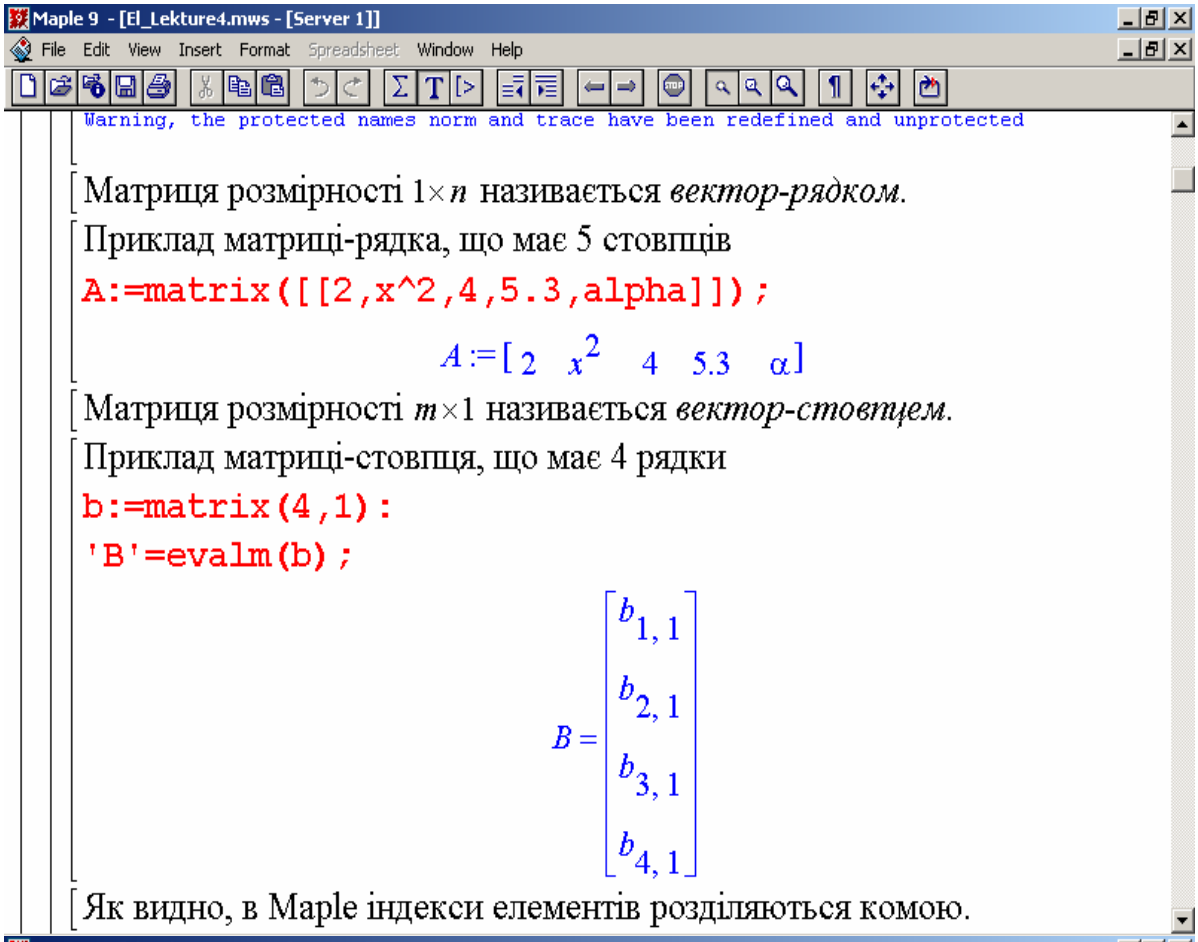
Стовпці матриці A позначають через A_j, j=1, 2, ..., n. Тоді матриця може бути записана так:

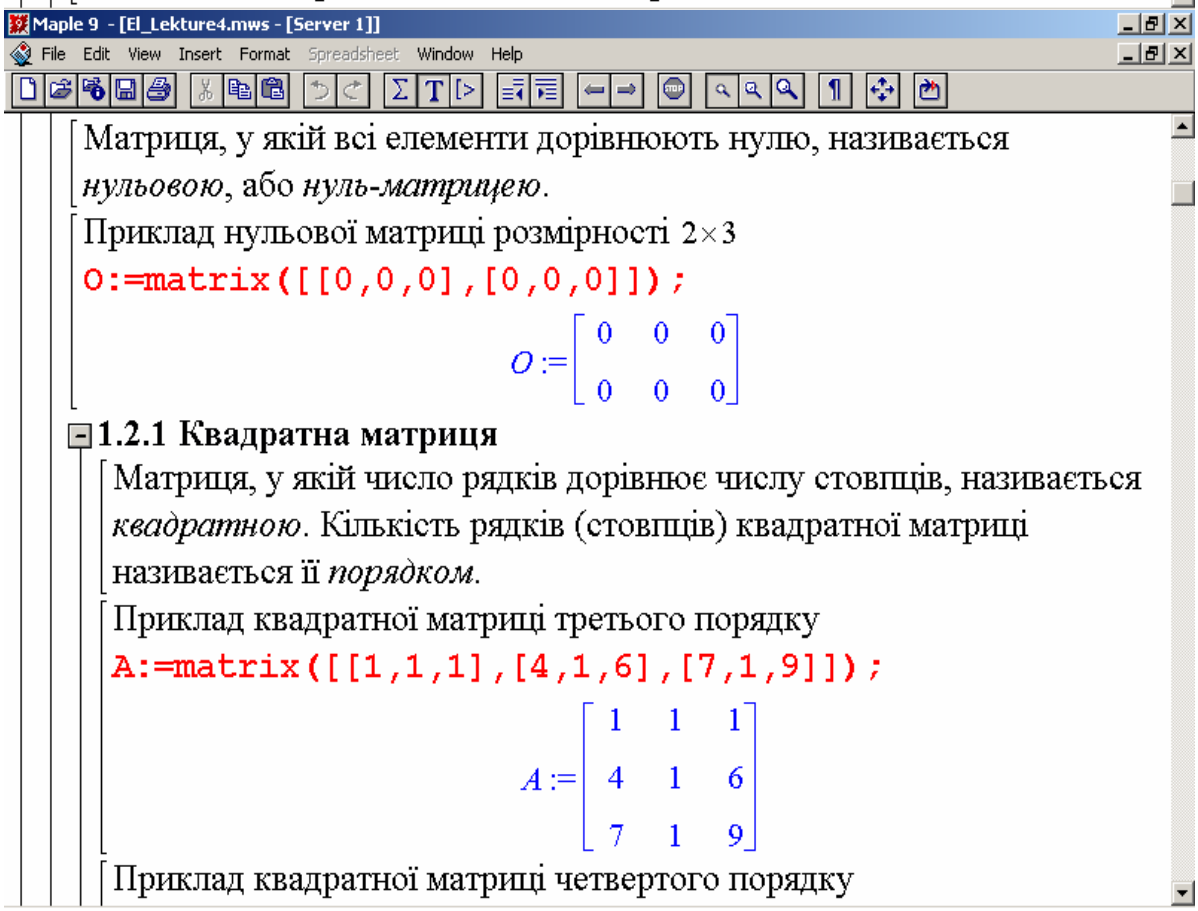
$A=(A_1, A_2, \dots, A_n)$.

Для задання матриці в Maple використовується команда **matrix**. Причому задавати матрицю можна різними способами, як видно з декількох прикладів, які приведені далі.

1.2 Види матриць

```
> restart:
  with(linalg):
```


Maple 9 - [El_Lekture4.mws - [Server 1]]
File Edit View Insert Format Spreadsheet Window Help
Warning, the protected names norm and trace have been redefined and unprotected
Матриця розмірності 1×n називається вектор-рядком.
Приклад матриці-рядка, що має 5 стовпців
A:=matrix([[2,x^2,4,5.3,alpha]]);
A := [2 x^2 4 5.3 α]
Матриця розмірності m×1 називається вектор-стовпцем.
Приклад матриці-стовпця, що має 4 рядки
b:=matrix(4,1):
'B'=evalm(b);
B = [b1,1 b2,1 b3,1 b4,1]
Як видно, в Maple індекси елементів розділяються комою.


Maple 9 - [El_Lekture4.mws - [Server 1]]
File Edit View Insert Format Spreadsheet Window Help
Матриця, у якій всі елементи дорівнюють нулю, називається нульовою, або нуль-матрицею.
Приклад нульової матриці розмірності 2×3
O:=matrix([[0,0,0],[0,0,0]]);
O := [0 0 0; 0 0 0]
1.2.1 Квадратна матриця
Матриця, у якій число рядків дорівнює числу стовпців, називається квадратною. Кількість рядків (стовпців) квадратної матриці називається її порядком.
Приклад квадратної матриці третього порядку
A:=matrix([[1,1,1],[4,1,6],[7,1,9]]);
A := [1 1 1; 4 1 6; 7 1 9]
Приклад квадратної матриці четвертого порядку

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

```
B:=matrix(4,4,[seq(k^2,k=1..16)]);
```

$$B := \begin{bmatrix} 1 & 4 & 9 & 16 \\ 25 & 36 & 49 & 64 \\ 81 & 100 & 121 & 144 \\ 169 & 196 & 225 & 256 \end{bmatrix}$$

Елементи a_{11}, a_{22}, ..., a_{nn} утворюють *головну діагональ* квадратної матриці А.

Квадратна матриця називається *діагональною*, якщо вона має вигляд

```
V1:=seq(a[k,k],k=1..3):
diag(V1);
```

$$\begin{bmatrix} a_{1,1} & 0 & 0 \\ 0 & a_{2,2} & 0 \\ 0 & 0 & a_{3,3} \end{bmatrix}$$

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

В діагональній матриці всі елементи, які стоять не на головній діагоналі, дорівнюють нулю.

Діагональна матриця, всі діагональні елементи якої рівні між собою, називається *скалярною матрицею*.

Приклад скалярної матриці 4 - го порядку (λ - будь-яке число)

```
diag(seq(lambda,i=1..4));
```

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

Скалярна матриця, всі діагональні елементи якої дорівнюють одиниці, називається одиничною і позначається символом *E*.

Приклад одиничної матриці 3 - го порядку

```
E:=diag(seq(1,i=1..3));
```

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

$$E := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

>

1. 3 Дії над матрицями

```
> resrart:
  with(linalg):
Warning, the protected names norm and trace have been redefined and unprotected
```

1.3.1. Множення матриці на число або числа на матрицю - результатом є матриця, елементами якої є добутки елементів даної матриці на це число.

```
> A:=matrix([[1,1,1],[4,1,6],[7,1,9],[-2,1,0]]);
  gamma*A=evalm(gamma*A);
  B:=matrix([[1,1,1,1],[-4,1,6,0]]);`3*B`=evalm(3*
  B);
  `B*3`=evalm(B*3);
```

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

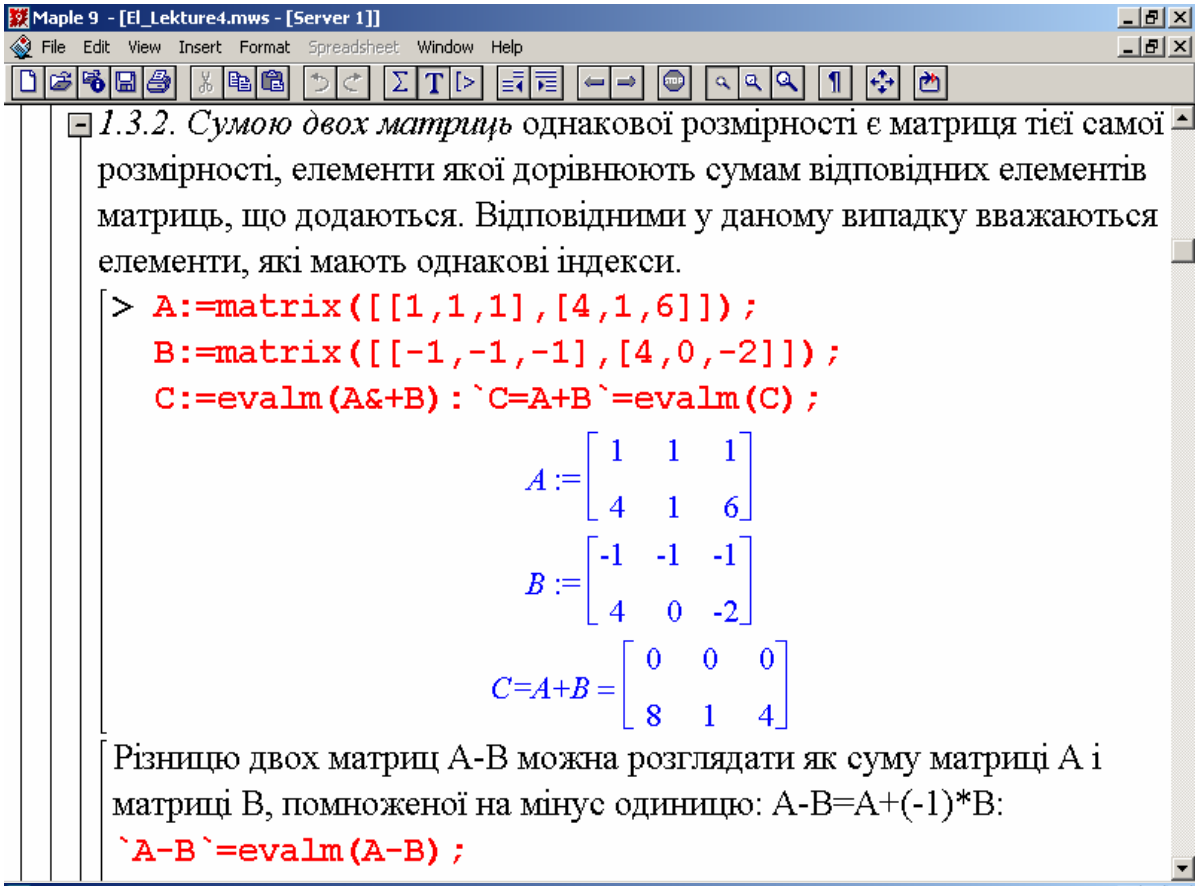
$$A := \begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \\ -2 & 1 & 0 \end{bmatrix}$$

$$\gamma A = \begin{bmatrix} \gamma & \gamma & \gamma \\ 4\gamma & \gamma & 6\gamma \\ 7\gamma & \gamma & 9\gamma \\ -2\gamma & \gamma & 0 \end{bmatrix}$$

$$B := \begin{bmatrix} 1 & 1 & 1 & 1 \\ -4 & 1 & 6 & 0 \end{bmatrix}$$

$$3*B = \begin{bmatrix} 3 & 3 & 3 & 3 \\ -12 & 3 & 18 & 0 \end{bmatrix}$$

$$B*3 = \begin{bmatrix} 3 & 3 & 3 & 3 \\ -12 & 3 & 18 & 0 \end{bmatrix}$$



1.3.2. Сумою двох матриць однакової розмірності є матриця тієї самої розмірності, елементи якої дорівнюють сумам відповідних елементів матриць, що додаються. Відповідними у даному випадку вважаються елементи, які мають однакові індекси.

```
> A:=matrix([[1,1,1],[4,1,6]]);
  B:=matrix([[-1,-1,-1],[4,0,-2]]);
  C:=evalm(A&+B):`C=A+B`=evalm(C);
```

$$A := \begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \end{bmatrix}$$

$$B := \begin{bmatrix} -1 & -1 & -1 \\ 4 & 0 & -2 \end{bmatrix}$$

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

Різницю двох матриц A-B можна розглядати як суму матриці A і матриці B, помноженої на мінус одиницю: A-B=A+(-1)*B:

```
`A-B`=evalm(A-B);
```

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

1.3.3. Транспонування матриці.

Якщо замість стовпців матриці A записати її рядки, зберігаючи їх порядок, тобто перший рядок записати першим стовпцем, другий рядок - другим стовпцем і т.д., то отримана матриця називається *транспонованою* до матриці A і позначається A^T.

Транспонування матрицы A виконується за допомогою команди **transpose**(A):

```
> A:=evalm(matrix([[0, 0, 0], [8, 1, 4]]));
  `transpose(A)`=transpose(A);
```

$$A := \begin{bmatrix} 0 & 0 & 0 \\ 8 & 1 & 4 \end{bmatrix}$$

$$transpose(A) = \begin{bmatrix} 0 & 8 \\ 0 & 1 \\ 0 & 4 \end{bmatrix}$$

Якщо замість рядків матриці A записати її стовпці, то отримаємо ту саму транспоновану матрицю. Якщо транспонувати A^T, то

отримаємо матрицю A.

```
> `transpose(transpose(A))`=transpose(transpose(A))
  );
```

$$transpose(transpose(A)) = \begin{bmatrix} 0 & 0 & 0 \\ 8 & 1 & 4 \end{bmatrix}$$

1.3.4. Множення матриць.

Операція множення матриць дещо складніша. Для зрозуміння суті цієї операції розглянемо приклад множення матриці - рядка на матрицю - стовпець:

```
> r:=4:m:=1:n:=1:
  a:=matrix(m,r):b:=matrix(r,n):
  `A`=evalm(a);`B`=evalm(b);
  print(`C=A*B`);C=evalm(a&*b);
```

$$A = [a_{1,1} \quad a_{1,2} \quad a_{1,3} \quad a_{1,4}]$$

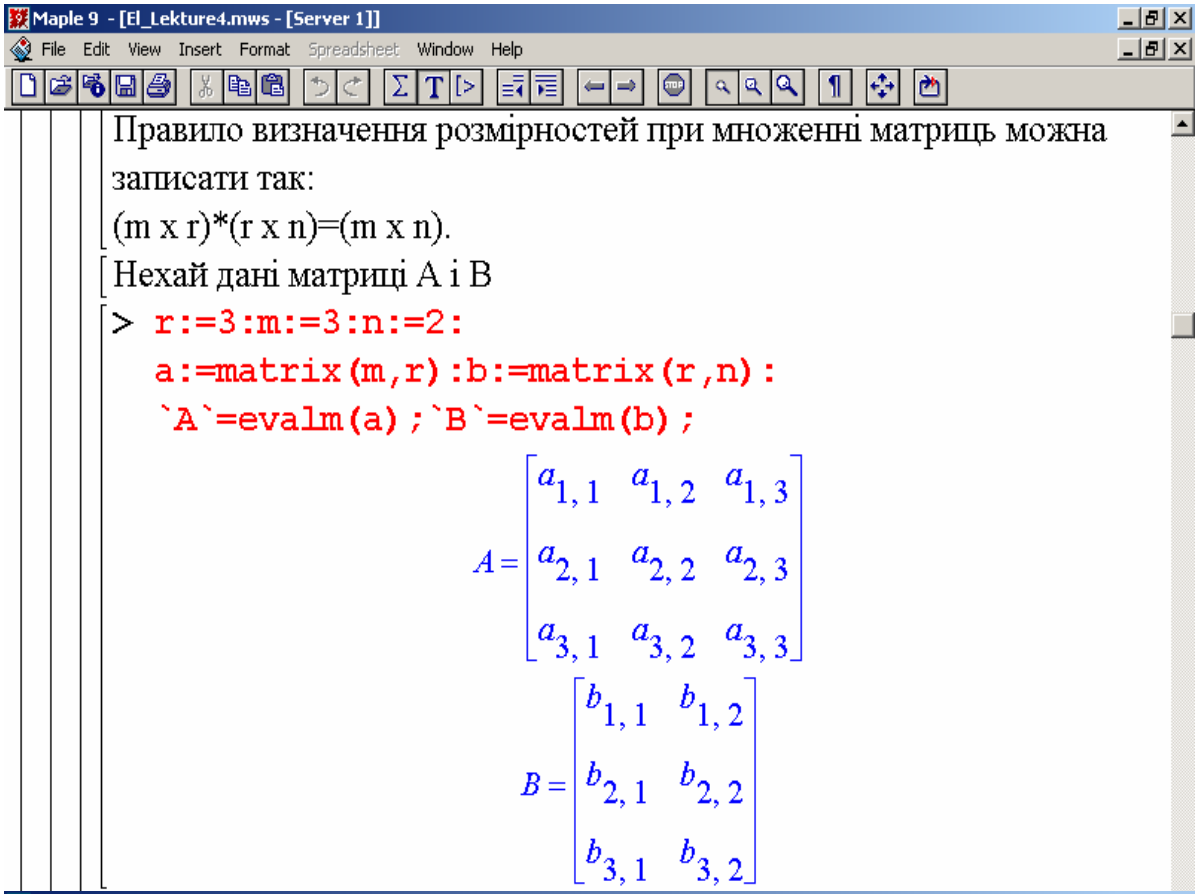
$$B = \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \\ b_{4,1} \end{bmatrix}$$

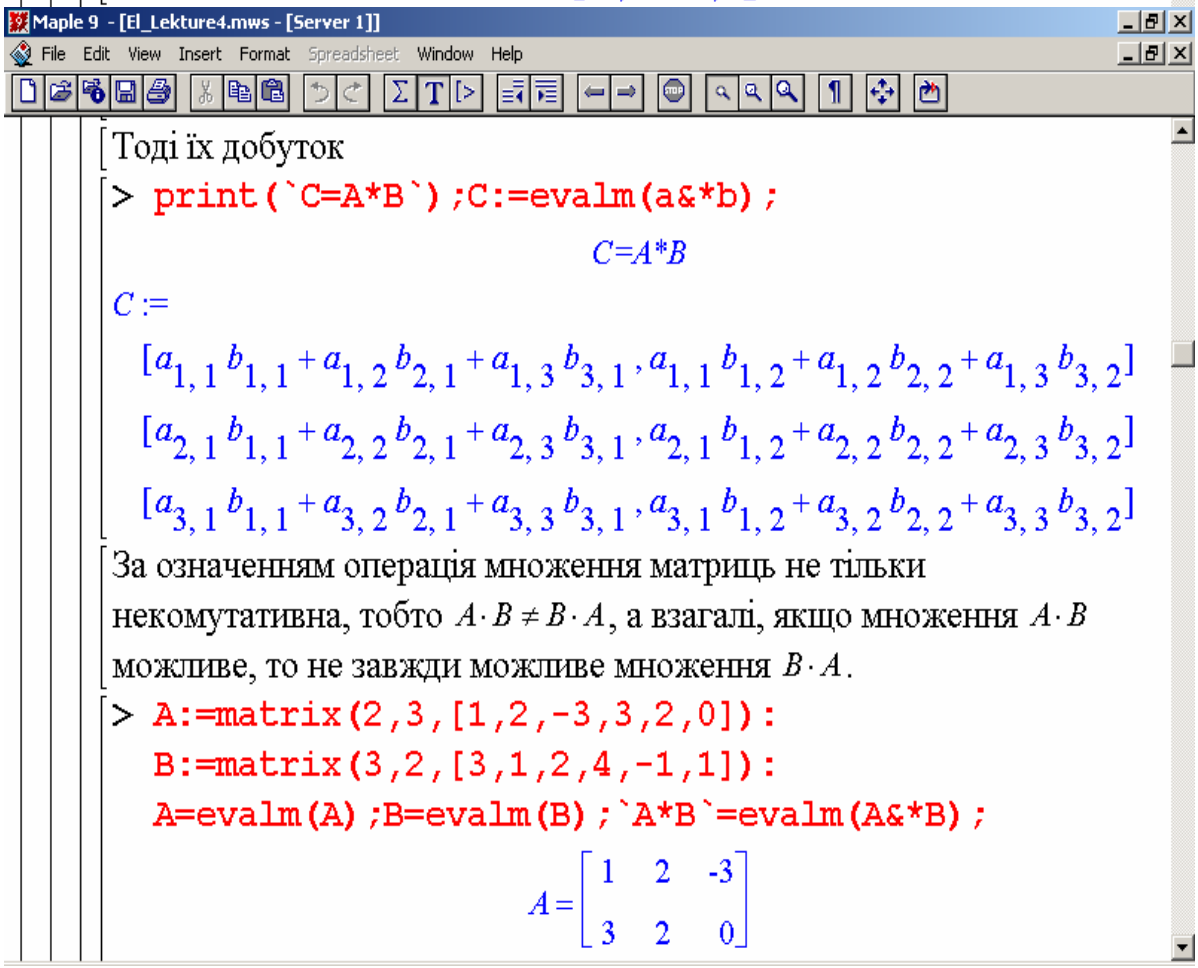
$$C = A*B$$

$$C = [a_{1,1}\,b_{1,1} + a_{1,2}\,b_{2,1} + a_{1,3}\,b_{3,1} + a_{1,4}\,b_{4,1}]$$

В приведеному прикладі матриця-рядок A розмірності $m \times r$ (m=1, r =4) множиться на матрицю-стовпець B розмірності $r \times n$ (n=1). Результатом є матриця C, розмірності $m \times n$, тобто 1×1. Єдиний елемент матриці C є сумою добутків елементів матриці A на відповідні елементи матриці B.

Операція добутку двох матриць A і B позначається так: $A \cdot B$. Матриці співмножники повинні задовольняти умову: кількість стовпців першої матриці повинна дорівнювати кількості рядків другої матриці. В результаті дістаємо матрицю C розмірності (m x n), де m - кількість рядків першої матриці, n - кількість стовпців другої матриці. Кожний елемент матриці c_{ij} дорівнює сумі добутків елементів i -го рядка матриці A на відповідні елементи j - го стовпця матриці B.


Maple 9 - [El_Lekture4.mws - [Server 1]]
File Edit View Insert Format Spreadsheet Window Help
Правило визначення розмірностей при множенні матриць можна записати так:
(m x r)*(r x n)=(m x n).
Нехай дані матриці A і B
> r:=3:m:=3:n:=2:
a:=matrix(m,r):b:=matrix(r,n):
`A`=evalm(a);`B`=evalm(b);
A = [a1,1 a1,2 a1,3; a2,1 a2,2 a2,3; a3,1 a3,2 a3,3]
B = [b1,1 b1,2; b2,1 b2,2; b3,1 b3,2]


Maple 9 - [El_Lekture4.mws - [Server 1]]
File Edit View Insert Format Spreadsheet Window Help
Тоді їх добуток
> print(`C=A*B`);C:=evalm(a&*b);
C=A*B
C :=
[a1,1 b1,1 + a1,2 b2,1 + a1,3 b3,1, a1,1 b1,2 + a1,2 b2,2 + a1,3 b3,2]
[a2,1 b1,1 + a2,2 b2,1 + a2,3 b3,1, a2,1 b1,2 + a2,2 b2,2 + a2,3 b3,2]
[a3,1 b1,1 + a3,2 b2,1 + a3,3 b3,1, a3,1 b1,2 + a3,2 b2,2 + a3,3 b3,2]
За означенням операція множення матриць не тільки некомутативна, тобто A·B ≠ B·A, а взагалі, якщо множення A·B можливе, то не завжди можливе множення B·A.
> A:=matrix(2,3,[1,2,-3,3,2,0]):
B:=matrix(3,2,[3,1,2,4,-1,1]):
A=evalm(A);B=evalm(B);`A*B`=evalm(A&*B);
A = [1 2 -3; 3 2 0]

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

$$A*B = \begin{bmatrix} 10 & 6 \\ 13 & 11 \end{bmatrix}$$

Спробуємо знайти добуток $B \cdot A$ для заданих вище матриць

```
> `B*A`=evalm(b&*c);
```

$$B*A = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix} \&* c$$

"помилка, невідповідність розмірності для операції множення матриць".

Щоб взнати кількість рядків або стовпців матриці необхідно виконати такі операції:


```
> 'A'=evalm(A);
  `кількість рядків матриці A`=rowdim(A);
  `кількість стовпців матриці A`=coldim(A);
  print(`Розмірність матриці A`);
  matrix(1,3,[linalg[rowdim](A),`*`,linalg[coldim]
  (A)]);
```

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

$$\textit{кількість рядків матриці } A = \text{rowdim}(A)$$

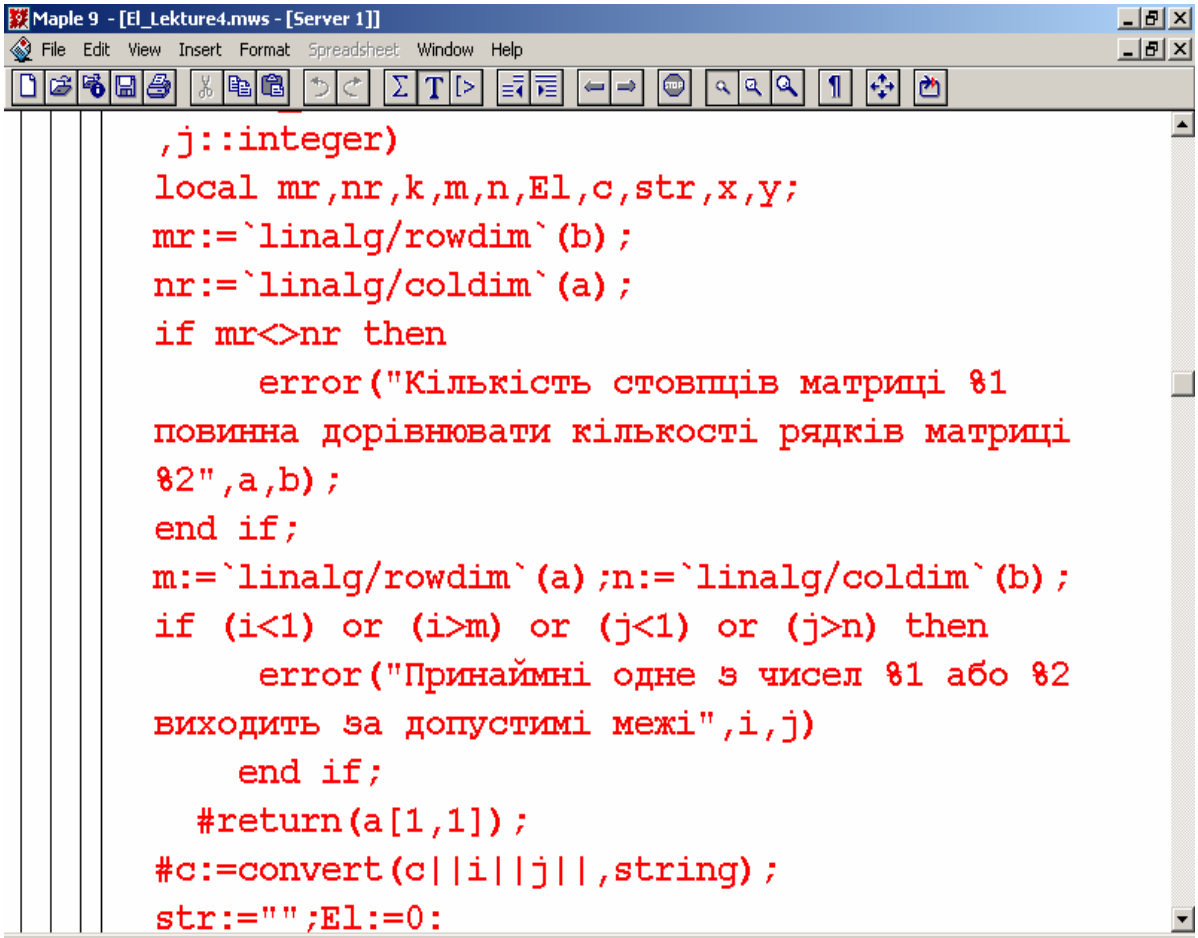
$$\textit{кількість стовпців матриці } A = \text{coldim}(A)$$

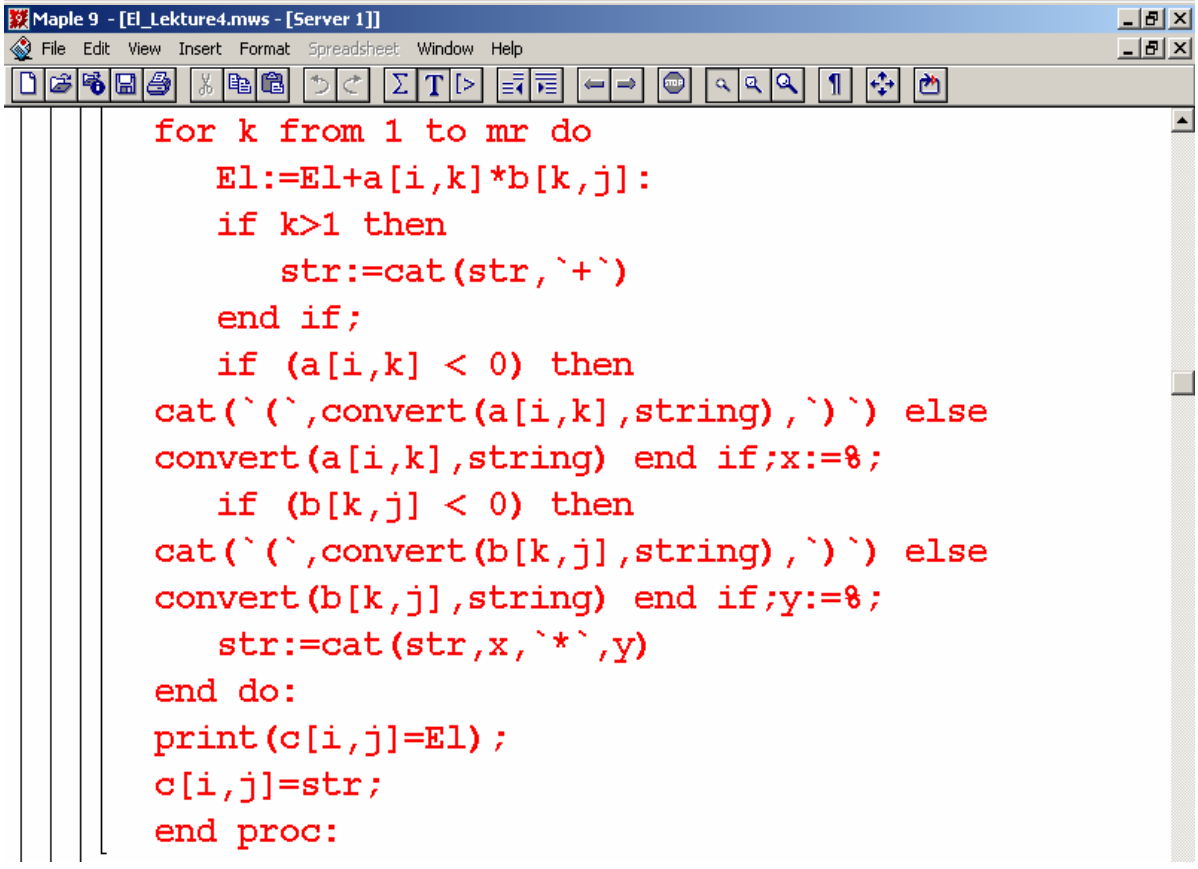
$$\textit{Розмірність матриці } A$$

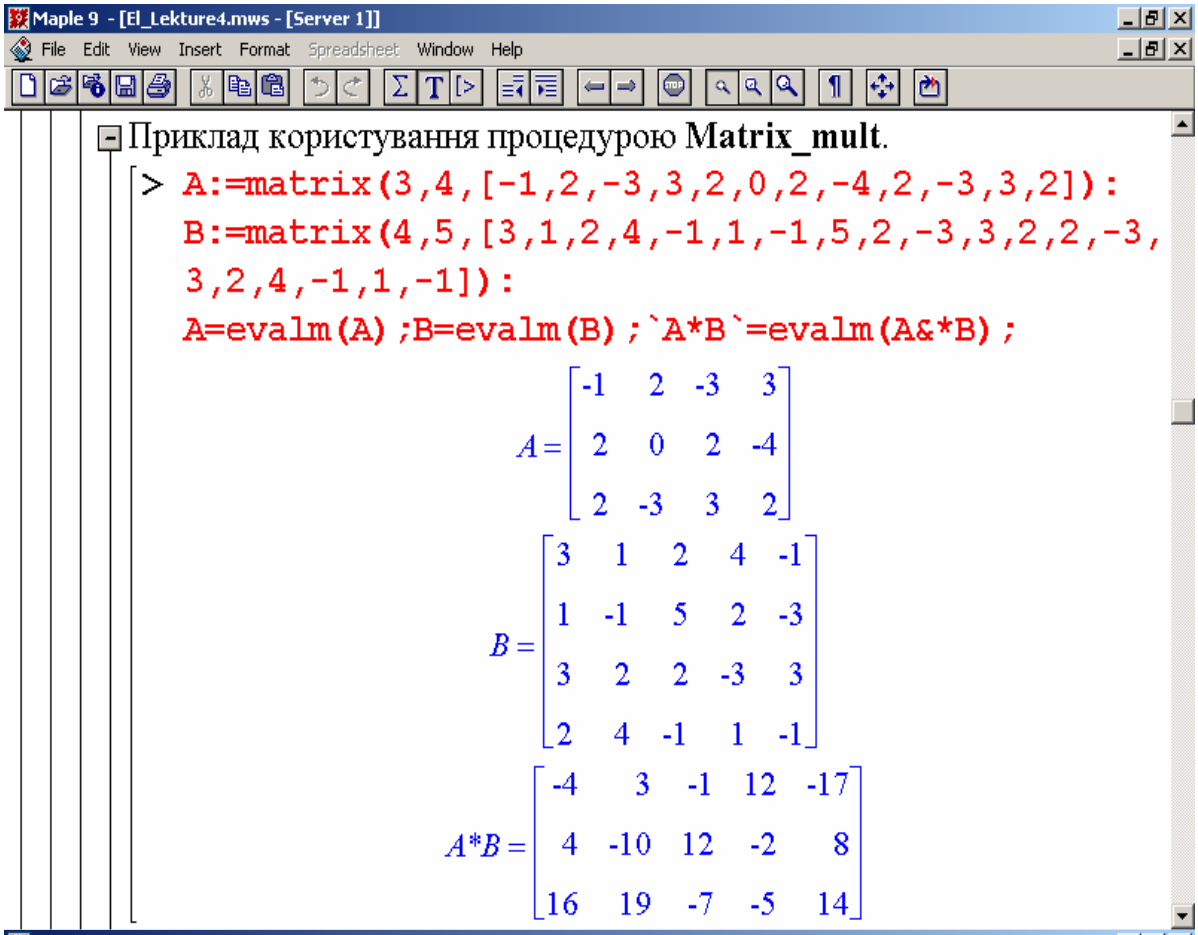
$$[2 \quad * \quad 3]$$

Приведена нижче спеціально створена процедура дозволяє для будь-якого елемента матриці C (C=$A \cdot B$) отримати детальний опис його обчислення через елементи матриці A та B.

```
> Matrix_mult:=proc(a::matrix,b::matrix,i::integer
```


Maple 9 - [El_Lekture4.mws - [Server 1]]
File Edit View Insert Format Spreadsheet Window Help
,j::integer)
local mr,nr,k,m,n,El,c,str,x,y;
mr:=`linalg/rowdim`(b);
nr:=`linalg/coldim`(a);
if mr<>nr then
error("Кількість стовпців матриці %1
повинна дорівнювати кількості рядків матриці
%2",a,b);
end if;
m:=`linalg/rowdim`(a);n:=`linalg/coldim`(b);
if (i<1) or (i>m) or (j<1) or (j>n) then
error("Принаймні одне з чисел %1 або %2
виходить за допустимі межі",i,j)
end if;
#return(a[1,1]);
#c:=convert(c||i||j||,string);
str:="";El:=0:


Maple 9 - [El_Lekture4.mws - [Server 1]]
File Edit View Insert Format Spreadsheet Window Help
for k from 1 to mr do
El:=El+a[i,k]*b[k,j]:
if k>1 then
str:=cat(str,`+`)
end if;
if (a[i,k] < 0) then
cat(`(`,convert(a[i,k],string),`)`) else
convert(a[i,k],string) end if;x:=%;
if (b[k,j] < 0) then
cat(`(`,convert(b[k,j],string),`)`) else
convert(b[k,j],string) end if;y:=%;
str:=cat(str,x,`*`,y)
end do:
print(c[i,j]=El);
c[i,j]=str;
end proc:



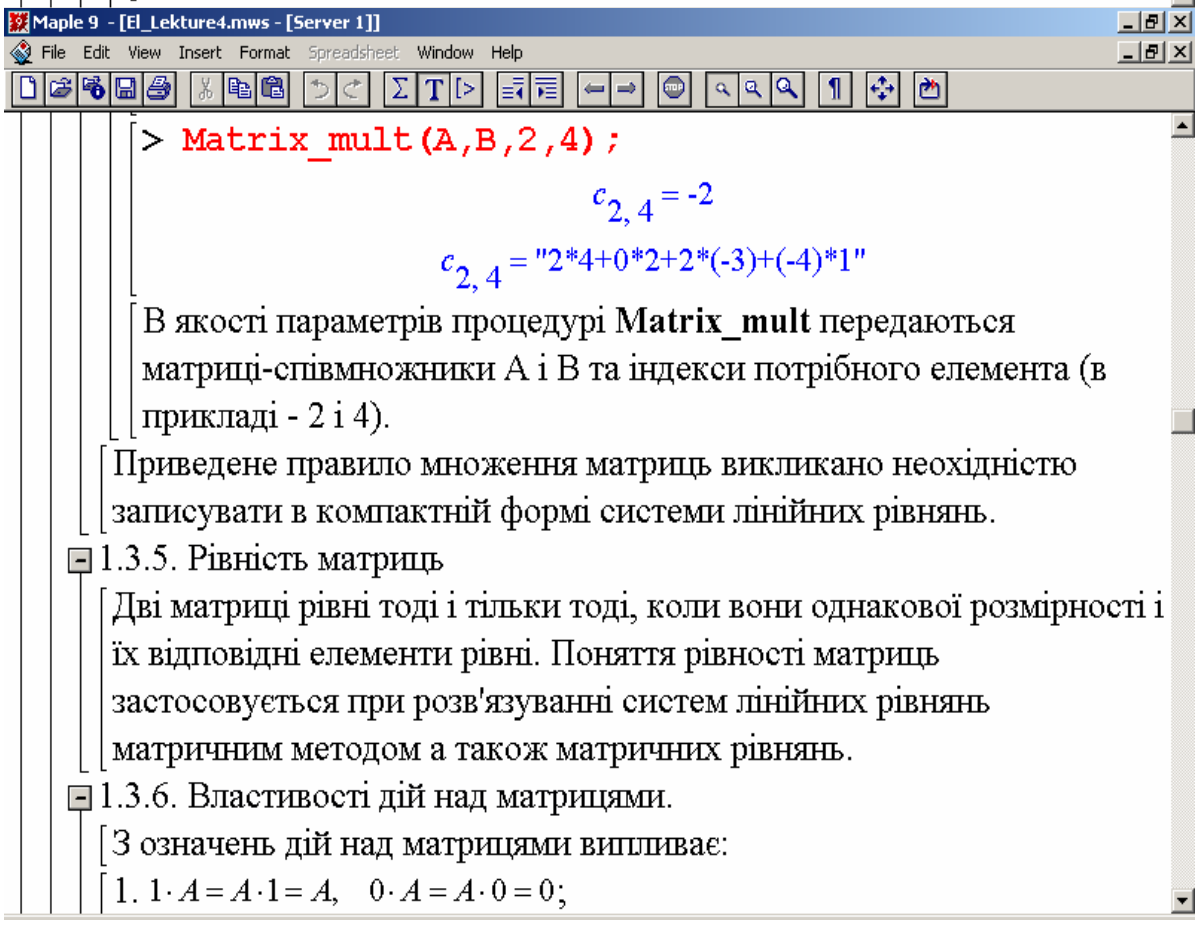
Приклад користування процедурою **Matrix_mult**.

```
> A:=matrix(3,4,[-1,2,-3,3,2,0,2,-4,2,-3,3,2]):
  B:=matrix(4,5,[3,1,2,4,-1,1,-1,5,2,-3,3,2,2,-3,
  3,2,4,-1,1,-1]):
  A=evalm(A);B=evalm(B);`A*B`=evalm(A&*B);
```

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

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

$$A*B=\begin{bmatrix}-4 & 3 & -1 & 12 & -17\\ 4 & -10 & 12 & -2 & 8\\ 16 & 19 & -7 & -5 & 14\end{bmatrix}$$




```
> Matrix_mult(A,B,2,4);
```

$$c_{2,4}=-2$$

$$c_{2,4}="2*4+0*2+2*(-3)+(-4)*1"$$

В якості параметрів процедурі **Matrix_mult** передаються матриці-співмножники A і B та індекси потрібного елемента (в прикладі - 2 і 4).

Приведене правило множення матриць викликано неохідністю записувати в компактній формі системи лінійних рівнянь.

1.3.5. Рівність матриць

Дві матриці рівні тоді і тільки тоді, коли вони однакової розмірності і їх відповідні елементи рівні. Поняття рівності матриць застосовується при розв'язуванні систем лінійних рівнянь матричним методом а також матричних рівнянь.

1.3.6. Властивості дій над матрицями.

З означень дій над матрицями випливає:

1. $1\cdot A=A\cdot 1=A,\quad 0\cdot A=A\cdot 0=0;$

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

2. $\alpha \cdot 0 = 0 \cdot \alpha = 0$;

3. $\alpha \cdot (\beta \cdot A) = (\alpha \cdot \beta) \cdot A = (A \cdot \alpha) \cdot \beta = A \cdot (\alpha \cdot \beta)$;

4. A+B=B+A ;

5. A+(B+C)=(A+B)+C;

6. (α+β)A=αA+βΦ, A(α+β)=Aα+Aβ ;

7. A+0=0+A=A;

8. α(AB)=(αA)B;

9. (A+B)C=AC+BC, C(A+B)=CA+CB;

10. A(BC)=(AB)C;

11 Якщо A - квадратна матриця, а E - одинична, то $AE=EA=A$;

Операція множення матриць в загальному випадку некомутативна, тобто AB≠BA. Для окремих матриць можлива рівність AB=BA.

1. 4 Матрична форма запису системи лінійних рівнянь

Система n лінійних рівнянь записується так

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m. \end{cases} \qquad (1)$$

Нехай

```
restart:
m:=4:n:=3:
a:=matrix(m,n):A=evalm(a);
X:=Vector([seq(x[k],k=1..n)],readonly=true);
B:=Vector([seq(b[k],k=1..n)],readonly=true);
```

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \\ a_{4,1} & a_{4,2} & a_{4,3} \end{bmatrix}$$

$$X := \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

$$B := \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Знайдемо добуток матриць AX:

```
> Ax:=evalm(a&*X):Ax:=convert(Ax,matrix):`AX`=evalm(
  Ax);
```

$$AX = \begin{bmatrix} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 \\ a_{2,1}x_1 + a_{2,2}x_2 + a_{2,3}x_3 \\ a_{3,1}x_1 + a_{3,2}x_2 + a_{3,3}x_3 \\ a_{4,1}x_1 + a_{4,2}x_2 + a_{4,3}x_3 \end{bmatrix}$$

Добуток матриці A на матрицю X, який запям'ятовано під ім'ям Ax, є матрицею розмірності $m \times 1$, такої самої, як і матриці B. Отже, можемо окремо викликати, наприклад, перші елементи матриць Ax та B:

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

```
> Ax[1,1];B[1];
```

$$a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3$$

$$b_1$$

Перший вираз є лівою частиною першого рівняння в системі лінійних рівнянь (1), а другий вираз - правою частиною. Отже, перше рівняння системи можна записати, як рівність відповідних елементів матриць Ax та B:

```
> Ax[1,1]=B[1];
```

$$a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 = b_1$$

Компактно всі рівняння системи можна записати як рівність матриць:

AX=B

2 Визначники

2.1 Поняття визначника матриці другого порядку

Якщо

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

$$A=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix},$$

то число $a_{11}\cdot a_{22}-a_{12}\cdot a_{21}$ називається ***визначником*** (наголос на виділеному складі ***на***) матриці А. Позначається *визначник* так

$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}$ або detA, ще |A|, іноді, як $\det(A_1,A_2)$, де A_1 - перший стовпець матриці А, A_2 - другий стовпець.

```
> resrart:
  with(linalg):
```

```
Warning, the protected names norm and trace have been redefined and unprotected
```

Для обчислення визначника використовується команда **det.** Формулу обчислення визначника в символьному вигляді можна отримати так

```
> m:=2:a:=matrix(m,m):`A`=evalm(a);
  `det(A)`=det(a);
```

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

$$A=\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}$$

$$det(A)=a_{1,1}\,a_{2,2}-a_{1,2}\,a_{2,1}$$

2.2 Властивості визначника матриці другого порядку

1. Визначник не змінюється при транспонуванні матриці:

$\det(A)=\det(A^T)$.

2. Визначник міняє знак, якщо поміняти місцями два стовпці (рядки):

$\det(A_1,A_2)=-\det(A_2,A_1)$.

3. Визначник дорівнює нулю, якщо він має однакові стовпці (рядки), тобто при $A_1=A_2$ виконується рівність: $\det(A_1,A_2)=0$.

Ця властивість має просте та красиве доведення:

З одного боку,

$$\det(A_1,A_2)=\det(A_2,A_1),$$

оскільки $A_1=A_2$. З іншого боку, на підставі властивості 2

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

$$-\det(A_1, A_2) = \det(A_2, A_1).$$

Звідси випливає, що

$$\det(A_1, A_2) = -\det(A_1, A_2),$$

що можливо лише коли $\det(A_1, A_2) = 0$.

4. Спільний множник елементів стовпця (рядка) можна виносити за знак визначника:

$$\det(\alpha \cdot A_1, A_2) = \det(A_1, \alpha \cdot A_2) = \alpha \cdot \det(A_1, A_2).$$

Наслідок 1: $\det(\alpha \cdot A) = \alpha^2 \cdot \det(A)$.

Наслідок 2: Якщо всі елементи стовпця (рядка) матриці дорівнюють нулю, то визначник матриці дорівнює нулю.

5. Якщо кожен елемент першого стовпця (рядка) є сумою двох доданків, то визначник дорівнює сумі двох визначників, у яких першими стовпцями (рядками) є відповідні доданки, а другий стовпець (рядок) є тим самим (те ж саме справедливо і для другого стовпця (рядка):

$$\det(A_1' + A_1'', A_2) = \det(A_1', A_2) + \det(A_1'', A_2).$$

6. Властивості 5, 6 у сукупності дають властивість лінійності

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

визначника:

$$\det(\alpha' \cdot A_1' + \alpha'' \cdot A_1'', A_2) = \alpha' \cdot \det(A_1', A_2) + \alpha'' \cdot \det(A_1'', A_2).$$

7. Визначник не зміниться, якщо до елементів будь-якого його стовпця (рядка) додати відповідні елементи іншого стовпця (рядка), помножені на одне й те саме число:

$$\det(A_1 + c_2 \cdot A_2, A_2) = \det(A_1, A_2 + c_1 \cdot A_1) = \det(A_1, A_2).$$

Ця властивість легко доводиться на основі властивості лінійності визначника і широко застосовується для обчислення визначників матриць третього та вищих порядків.

2.3 Поняття та властивості визначника матриці третього порядку

Якщо матриця

$$A = (A_1, A_2, A_3) = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix},$$

то число $a_{1,1}\,a_{2,2}\,a_{3,3} - a_{1,1}\,a_{2,3}\,a_{3,2} - a_{2,1}\,a_{1,2}\,a_{3,3} + a_{2,1}\,a_{1,3}\,a_{3,2} + a_{3,1}\,a_{1,2}\,a_{2,3} - a_{3,1}\,a_{1,3}\,a_{2,2}$ називається визначником матриці A.

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

Властивості визначника матриці третього порядку аналогічні властивостям визначника матриці другого порядку, тому запишемо ці властивості:

1. Визначник не змінюється при транспонуванні матриці: detA=detA^T

2. $\det(A_1,A_2,A_3)=-\det(A_3,A_2,A_1)=\det(A_2,A_1,A_3)=-\det(A_1,A_3,A_2)$.
3. $\det(A_1,A_1,A_3)=\det(A_1,A_2,A_2)=\det(A_1,A_2,A_1)=0$.
4. $\det(\alpha\cdot A_1,A_2,A_3)=\alpha\cdot\det(A_1,A_2,A_3)$.

   Наслідок: $\det(\alpha\cdot A)=\alpha^3\cdot\det(A)$.
5. $\det(A_1'+A_1'',A_2,A_3)=\det(A_1',A_2,A_3)+\det(A_1'',A_2,A_3)$.
6. $\det(\alpha'\cdot A_1'+\alpha''\cdot A_1'',A_2,A_3)=\alpha'\cdot\det(A_1',A_2,A_3)+\alpha''\cdot\det(A_1'',A_2,A_3)$.
7. $\det(A_1+c_2\cdot A_2,A_2,A_3)=\det(A_1,A_2+c_1\cdot A_1,A_3)=\det(A_1+c_3\cdot A_3,A_2,A_3)=$
   $=\det(A_1,A_2,A_3)$.

(Самостійна робота: записати відповідні означення до приведених властивостей)

Формулу обчислення визначника в символьному вигляді можна

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

отримати так

```
> m:=3:a:=matrix(m,m):`A`=evalm(a);
  `det(A)`=det(a);
```

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

$$det(A)=a_{1,1}a_{2,2}a_{3,3}-a_{1,1}a_{2,3}a_{3,2}-a_{2,1}a_{1,2}a_{3,3}+a_{2,1}a_{1,3}a_{3,2}+a_{3,1}a_{1,2}a_{2,3}-a_{3,1}a_{1,3}a_{2,2}$$

2.4 Перевірка властивостей визначника

Для перевірки властивостей визначника скористаємося командами:

Властивість 1: Визначник не змінюється при транспонуванні матриці: detA=detA^T.

```
> A:=matrix([[1,1,1],[4,1,6],[7,1,9]]):
  print('A'=evalm(A),`transpose(A)`=transpose(A));
```

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

```
'det(A)-det(transpose(A))'=det(A)-det(transpose(A));
```

$$A=\begin{bmatrix}1&1&1\\4&1&6\\7&1&9\end{bmatrix},\ transpose(A)=\begin{bmatrix}1&4&7\\1&1&1\\1&6&9\end{bmatrix}$$

$$\det(A)-\det(\text{transpose}(A))=0$$

Властивість 2: Визначник міняє знак, якщо поміняти місцями два стовпці (рядки).

```
> A:=matrix([[1,1,1],[4,1,6],[7,1,9]]):
print('A'=evalm(A),`detA`=det(A));
print(`В матриці А поміняли місцями перший і третій рядки`);
`B`=swaprow(A,1,3):#
```

Міняються місцями перший і третій рядки

```
print(evalm(%),`detB`=det(rhs(%)));
```

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

```
print(`В матриці А поміняли місцями перший і другий стовпці`);
`C`=swapcol(A,2,1):#
```

Міняються місцями перший і другий стовпці

```
> print(evalm(%),`detC`=det(rhs(%)));
```

$$A=\begin{bmatrix}1&1&1\\4&1&6\\7&1&9\end{bmatrix},\ detA=6$$

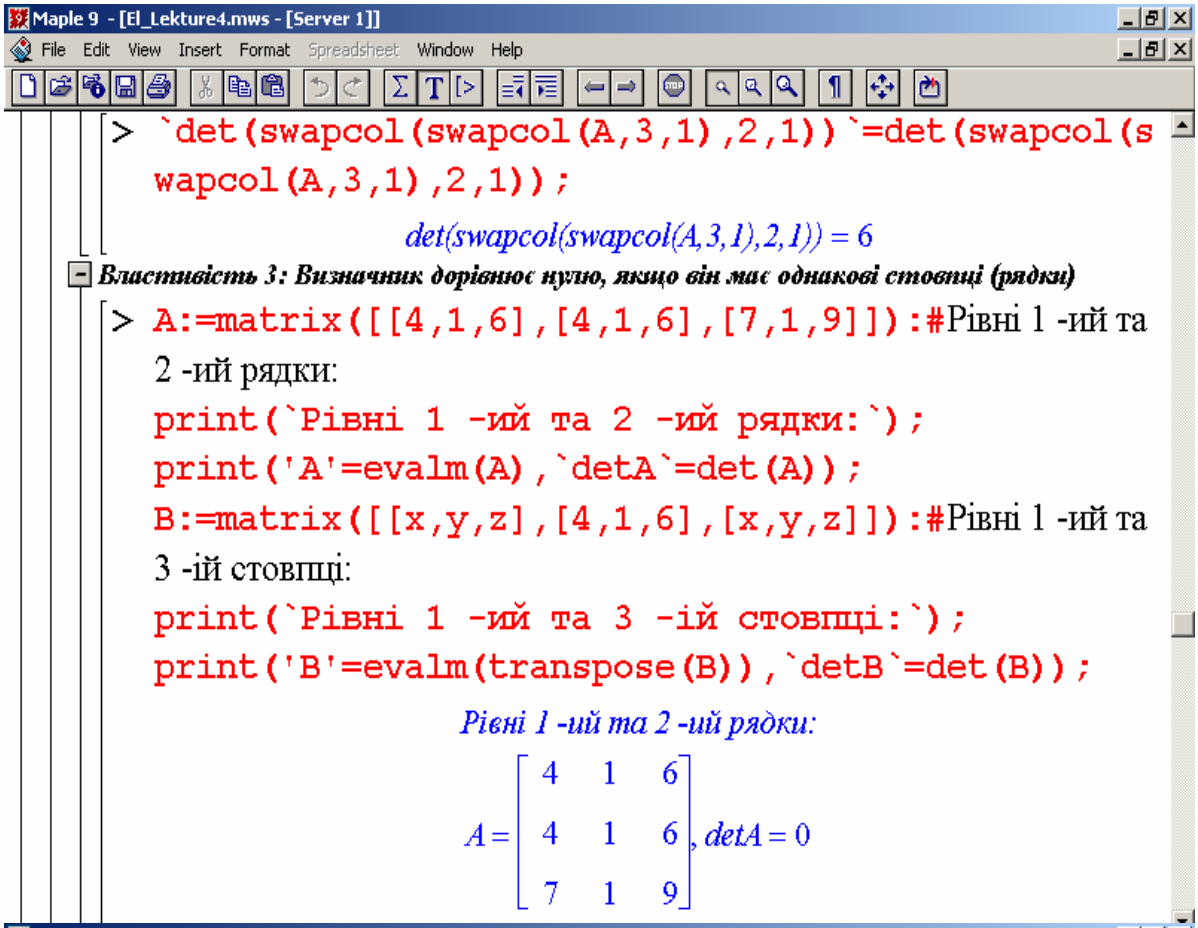
В матриці А поміняли місцями перший і третій рядки

$$B=\begin{bmatrix}7&1&9\\4&1&6\\1&1&1\end{bmatrix},\ detB=-6$$

В матриці А поміняли місцями перший і другий стовпці

$$C=\begin{bmatrix}1&1&1\\1&4&6\\1&7&9\end{bmatrix},\ detC=-6$$

Природно, що при подвійній перестановці стовпців знак визначника не змінюється




```
> `det(swapcol(swapcol(A,3,1),2,1))`=det(swapcol(swapcol(A,3,1),2,1));
```

$$det(swapcol(swapcol(A,3,1),2,1)) = 6$$

Властивість 3: Визначник дорівнює нулю, якщо він має однакові стовпці (рядки)

```
> A:=matrix([[4,1,6],[4,1,6],[7,1,9]]):#Рівні 1 -ий та 2 -ий рядки:
  print(`Рівні 1 -ий та 2 -ий рядки:`);
  print('A'=evalm(A),`detA`=det(A));
  B:=matrix([[x,y,z],[4,1,6],[x,y,z]]):#Рівні 1 -ий та 3 -ій стовпці:
  print(`Рівні 1 -ий та 3 -ій стовпці:`);
  print('B'=evalm(transpose(B)),`detB`=det(B));
```

Рівні 1 -ий та 2 -ий рядки:

$$A = \begin{bmatrix} 4 & 1 & 6 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \end{bmatrix}, detA = 0$$

Рівні 1 -ий та 3 -ій стовпці:

$$B = \begin{bmatrix} x & 4 & x \\ y & 1 & y \\ z & 6 & z \end{bmatrix}, detB = 0$$

Властивість 4: Спільний множник елементів стовпця (рядка) можна виносити за знак визначника:

Помножити стовпець з номером j на значення виразу *expr* можна за допомогою команди mulcol(A,*j*,*expr*), а та ж сама дія для рядка з номером *i* виконується командою mulrow(A,*i*,*expr*).

```
> A:=matrix([[1,1,1],[4,1,6],[7,1,9]]):
  print(`A`=evalm(A),'det(A)'=det(A));
  alpha:='alpha':
  print(`mulcol(A,1,alpha)`=mulcol(A,1,alpha),'det(mulcol(A,1,alpha))'=det(mulcol(A,1,alpha)));
```

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \end{bmatrix}, \det(A) = 6$$

$$mulcol(A,1,alpha) = \begin{bmatrix} \alpha & 1 & 1 \\ 4\,\alpha & 1 & 6 \\ 7\,\alpha & 1 & 9 \end{bmatrix}, \det(\mathrm{mulcol}(A, 1, \alpha)) = 6\,\alpha$$

Перевірка співвідношення $\det(\alpha \cdot A) = \alpha^n \cdot \det(A)$, де n - порядок матриці A.

```
> alpha:='alpha':
  print(`Порядок матриці n`=rowdim(A));
  print(`A`=evalm(A),'det(A)'=det(A));
  B:=evalm(alpha*A):
  print('B=alpha*A','B'=evalm(B));
  print('det(B)'=det(B));
```

Порядок матриці $n = 3$

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 1 & 6 \\ 7 & 1 & 9 \end{bmatrix}, \det(A) = 6$$

$$B = \alpha\,A, B = \begin{bmatrix} \alpha & \alpha & \alpha \\ 4\,\alpha & \alpha & 6\,\alpha \\ 7\,\alpha & \alpha & 9\,\alpha \end{bmatrix}$$

$$\det(B) = 6\,\alpha^3$$

Властивість 5: Якщо кожен елемент першого стовпця (рядка) є сумою двох доданків, то визначник дорівнює сумі двох визначників, у яких першими стовпцями (рядками) є відповідні доданки, а другий стовпець (рядок) є тим самим (те ж саме справедливо і для другого стовпця (рядка):

$$\det(A_1' + A_1'', A_2) = \det(A_1', A_2) + \det(A_1'', A_2).$$

```
> a:='a':
  A:=matrix([[a[1]+a[2],1,1],[b[1]+b[2],1,6],[c[1]
```

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

```
+c[2],1,9]]):
A1:=matrix([[a[1],1,1],[b[1],1,6],[c[1],1,9]]):
A2:=matrix([[a[2],1,1],[b[2],1,6],[c[2],1,9]]):
print('A'=evalm(A),'A1'=evalm(A1),'A2'=evalm(A2)
);
'det(A)-(det(A1)+det(A2))'=det(A)-(det(A1)+det(A
2));
```

$$A=\begin{bmatrix} a_1+a_2 & 1 & 1 \\ b_1+b_2 & 1 & 6 \\ c_1+c_2 & 1 & 9 \end{bmatrix}, A1=\begin{bmatrix} a_1 & 1 & 1 \\ b_1 & 1 & 6 \\ c_1 & 1 & 9 \end{bmatrix}, A2=\begin{bmatrix} a_2 & 1 & 1 \\ b_2 & 1 & 6 \\ c_2 & 1 & 9 \end{bmatrix}$$

$$\det(A)-\det(A1)-\det(A2)=0$$

Властивість 7: Визначник не зміниться, якщо до елементів будь-якого його стовпця (рядка) додати відповідні елементи іншого стовпця (рядка), помножені на одне й те саме число:

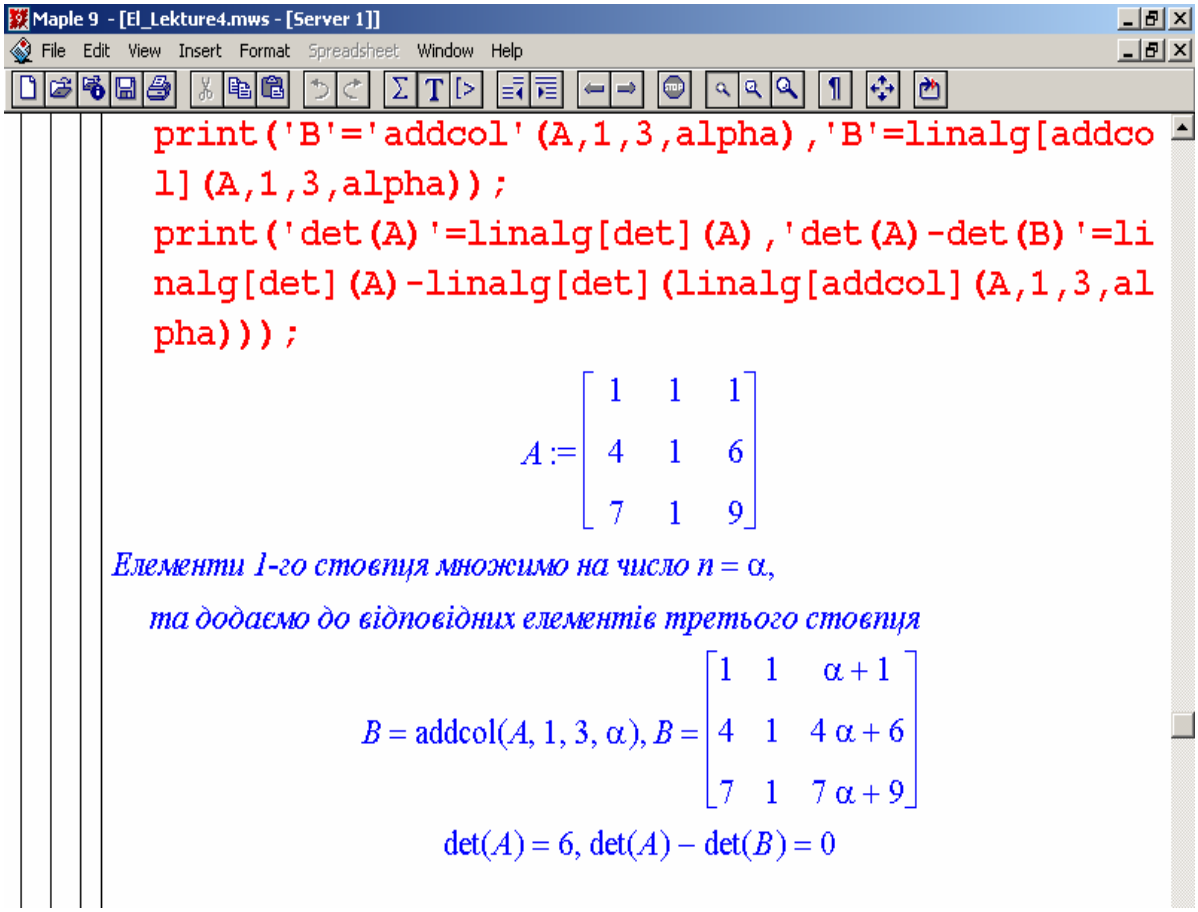
$\det(A_1+c_2\cdot A_2,A_2)=\det(A_1,A_2+c_1\cdot A_1)=\det(A_1,A_2)$.

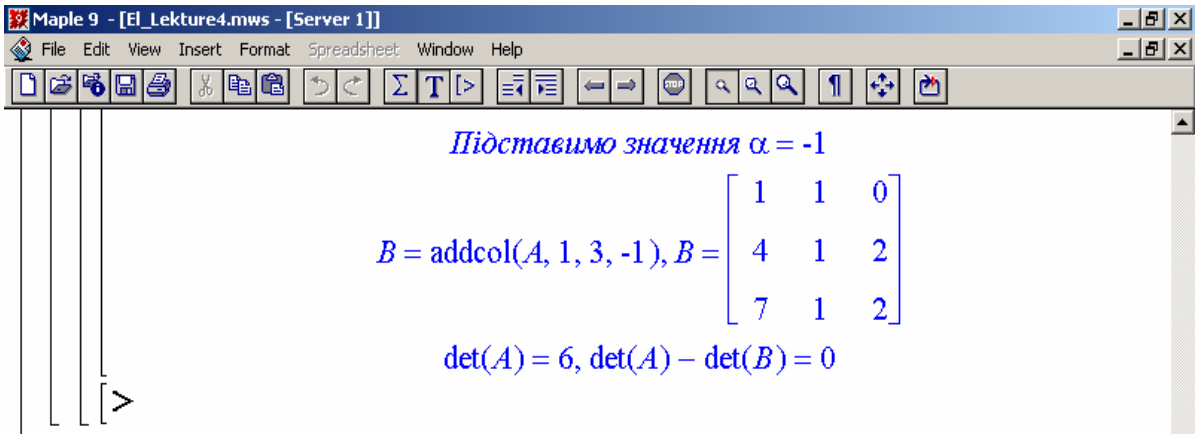
Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

Для перевірки цієї важливої властивості можна скористатись командами addrow(A,i,j,m) та addcol(A,i,j,m), за допомогою яких елементи *i-го* рядка (стовпця) множаться на скалярнийй множник **m** і додаються до відповідних елементів *j-го* рядка (стовпця).

```
> A:=matrix([[1,1,1],[4,1,6],[7,1,9]]);
  alpha:='alpha':
  print(`Елементи 1-го стовпця множимо на число
  n`=alpha,` та додаємо до відповідних елементів
  третього стовпця`);
  print('B'='addcol'(A,1,3,alpha),'B'=linalg[addco
  l](A,1,3,alpha));
  print('det(A)'=linalg[det](A),'det(A)-det(B)'=li
  nalg[det](A)-linalg[det](linalg[addcol](A,1,3,al
  pha)));
  alpha:=-1:
  print(``);
  print(`Підставимо значення`*'alpha'=alpha);
```





2.5 Обчислення визначника за теоремою розкладання

Означення 1. Нехай a_{ij} - довільний елемент квадратної матриці А порядку n. Якщо викреслити рядок і стовпець, у яких стоїть цей елемент, то дістанемо квадратну матрицю, на порядок меншу за початкову (тобто (n-1)-*го* порядку). Визначник цієї матриці називається мінором і позначається через M_{ij}.

Означення 2. Алгебраїчним доповненням називається мінор, помножений на число $(-1)^{(i+j)}$. Позначається алгебраїчне доповнення через A_{ij}. Отже,

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

$$A_{ij} = (-1)^{(i+j)} M_{ij}.$$

Теорема 1. (про розкладання визначника *n-го* порядку за елементами довільного рядка або стовпця). Якщо А- квадратна матриця, то її визначник дорівнює сумі добутків елементів будь-якого стовпця (рядка) на їх алгебраїчні доповнення.

(Без доведення).

```
> #Приклад
  n:=3:# Задання порядку матриці
  print(`Дана квадратна матриця порядку n`=n);
  a:=matrix(n,n):`A`=evalm(a);
  j:='j':i:='i':A:='A':
  `det(A)`=add(a[i,j]*'A'['i',j],j=1..coldim(a));
  print(`Або`);`det(A)`=det(a);
```

Дана квадратна матриця порядку $n = 3$

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

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

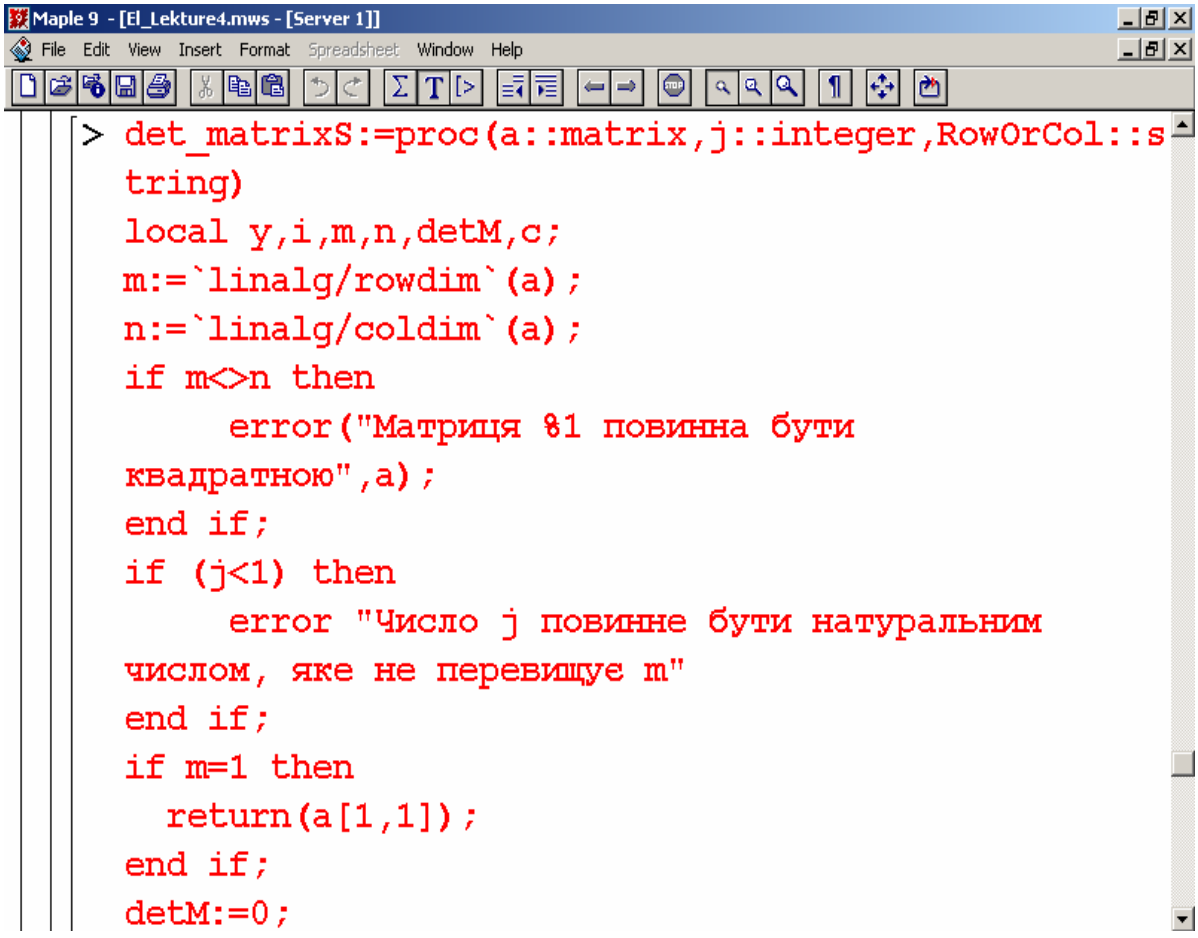
$$det(A) = a_{i,1} A_{i,1} + a_{i,2} A_{i,2} + a_{i,3} A_{i,3}$$

Або

$$det(A) = a_{1,1} a_{2,2} a_{3,3} - a_{1,1} a_{2,3} a_{3,2} - a_{2,1} a_{1,2} a_{3,3} + a_{2,1} a_{1,3} a_{3,2} + a_{3,1} a_{1,2} a_{2,3} - a_{3,1} a_{1,3} a_{2,2}$$

Згідно з теоремою розкладання, обчислення визначника будь-якого порядку зводиться до обчислення визначників 2- *го* порядку. Формули для обчислення визначників 2- *го* та 3- *го* порядків, що приведені у підрозділах 2.1, 2.3, випливають із теореми розкладання, як окремі випадки.

Для отримання символьного виду виразу для обчислення визначника за формулою розкладання може бути використана така процедура.

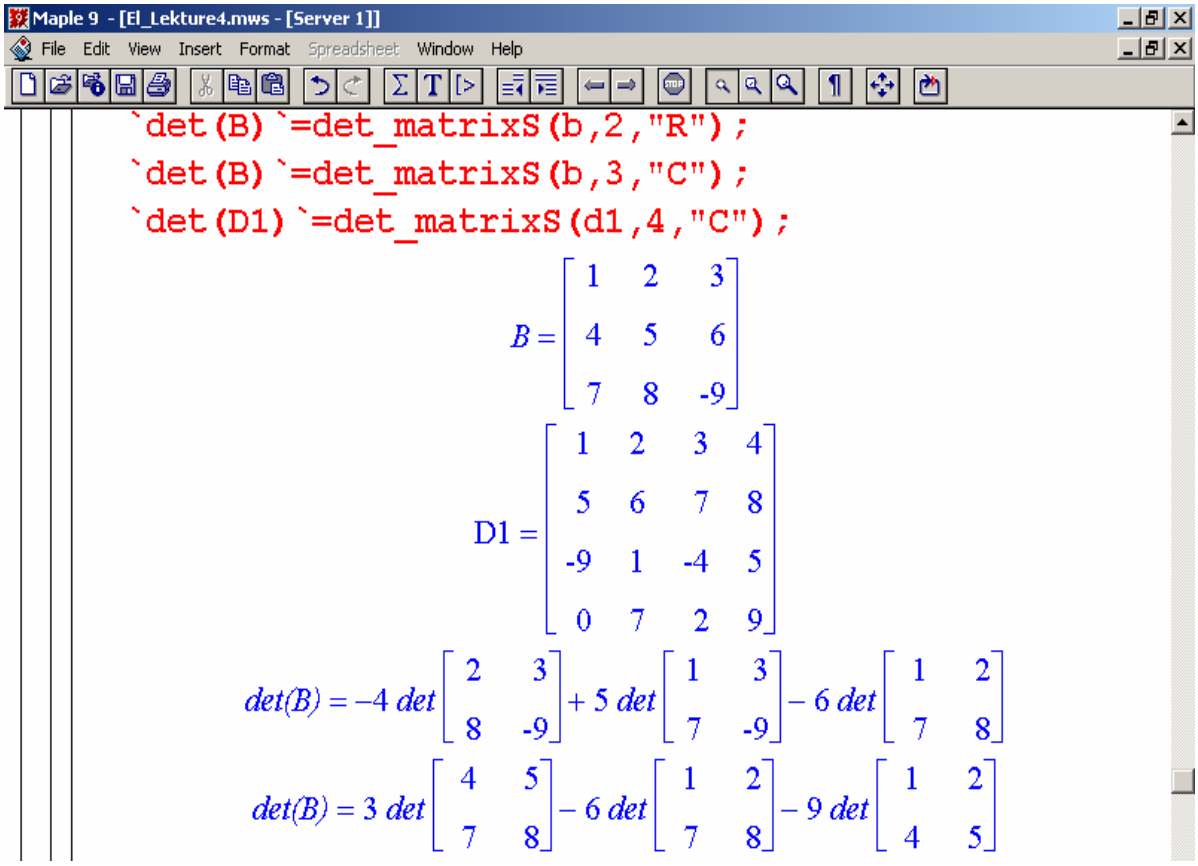


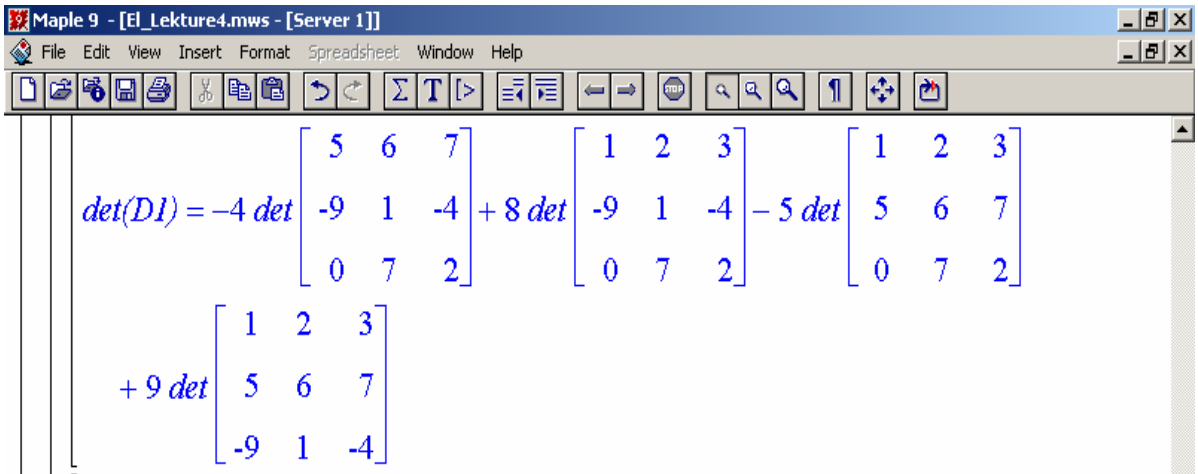

```
> det_matrixS:=proc(a::matrix,j::integer,RowOrCol::s
  tring)
  local y,i,m,n,detM,c;
  m:=`linalg/rowdim`(a);
  n:=`linalg/coldim`(a);
  if m<>n then
       error("Матриця %1 повинна бути
  квадратною",a);
  end if;
  if (j<1) then
       error "Число j повинне бути натуральним
  числом, яке не перевищує m"
  end if;
  if m=1 then
    return(a[1,1]);
  end if;
  detM:=0;
```


```
  if RowOrCol="R" then
  y:=sum('(-1)^(j+i)*a[j,i]*det*linalg['minor'](a,j,
  i)',
  'i'=1..m):
  else
  y:=sum('(-1)^(j+i)*a[i,j]*det*linalg['minor'](a,i,
  j)',
  'i'=1..m):
  end if
  end proc:
```

При виклику процедури необхідно вказати матрицю, номер ряду, по якому буде проведено розкладання, і вказати вид ряду, - горизонтальний ("R"-"Row") або вертикальний ("C" - "Column"):

```
> b:=matrix(3,3,[1,2,3,4,5,6,7,8,-9]):print(B=evalm(
  b));
  d1:=matrix(4,4,[1,2,3,4,5,6,7,8,-9,1,-4,5,0,7,2,9]
  ):print(D1=evalm(d1)):
```





Визначник - n - *го* порядку має властивості, аналогічні властивостям визначників 2- *го* та 3- *го* порядків.

2.6 Теорема про анулювання

Теорема. Сума добутків елементів будь-якого рядка (стовпця) визначника на алгебраїчні доповнення відповідних елементів іншого рядка (стовпця) визначника дорівнює нулю.

Доведення. Доведення будемо проводити на прикладі визначника 3- *го* порядку. Будемо розглядати суму добутків елмментів *i* - *го* рядка на алгебраїчні доповнення відповідних елементів *k* - *го* рядка матриці A,

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

тобто, потрібно довести рівність $\sum_{j=1}^{3} a_{i,j} A_{k,j} = 0 \ (i \neq k)$.

Нехай

```
> i:=1:k:=2:
  print('i'=i,'k'=k);
  print(`Тобто розглядатиме суму добутків елементів
  рядка номер i`=i);
  print(`визначника на алгебраїчні доповнення рядка
  номер k`=k);
  print(`Рівність, яку потрібно довести, приймає
  вигляд:`);
  Sum(a[i,j]*'A'[k,j],j=1..linalg[coldim](a))=0;
  print(`Або`);
  add(a[i,j]*'A'[k,j],j=1..linalg[coldim](a))=0;
```

$$i = 1, k = 2$$

Тобто розглядатиме суму добутків елементів рядка номер $i = 1$

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

визначника на алгебраїчні доповнення рядка номер $k = 2$

Рівність, яку потрібно довести, приймає вигляд:

$$\sum_{j=1}^{3} a_{1,j} A_{2,j} = 0$$

Або

$$a_{1,1} A_{2,1} + a_{1,2} A_{2,2} + a_{1,3} A_{2,3} = 0$$

Розглянемо дві матриці A та B.

```
> a:=matrix(n,n):B:=a:stackmatrix(row(a,i),B):
  B:=delrows(swaprow(%,1,k+1),1..1):
  print(`A`=evalm(a),`B`=evalm(B));
  print(`Ці матриці відрізняються тільки елементами
  рядка номер k`=k);
  print(`Значення лівої частини рівності:`);
  add(a[i,j]*'A'[k,j],j=1..coldim(a))=0;
  print(`записане для матриць A та B, не
```

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

```
змінюється,\n оскільки цей вираз не залежить від
елементів рядка номер k`=k);
print(`Дійсно:`);
l:='l':for l to coldim(a) do
        'A'[k,l]=det(minor(a,k,l))
       end do;
print(`Як видно, елементи матриці A, які стоять в
рядку номер k`=k,` тут відсутні.`);
print(`Інші ж лементи у матриць A та B - рівні`);
print(`Вираз:`*add(a[i,j]*'A'[k,j],
j=1..coldim(a)));
print(`Можна розглядати як визначник матриці B,\n
розписаний за елементами рядка номер k`=k);
print(`Але ж матриця B має два однакових рядки з
номерами`,('i'=i,'k'=k));
print(`За властивістю 3 визначників, для такої
матриці визначник дорівнює нулю.`);
```

Maple 9 - [El_Lekture4.mws - [Server 1]]

File Edit View Insert Format Spreadsheet Window Help

```
print(` Тобто повинна справджуватися рівність:
`);
add(a[i,j]*'A'[k,j],j=1..coldim(a))=0;
print(`Теорему доведено.`);
```

$$A=\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}, B=\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{1,1} & a_{1,2} & a_{1,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

Ці матриці відрізняються тільки елементами рядка номер $k=2$

Значення лівої частини рівності:

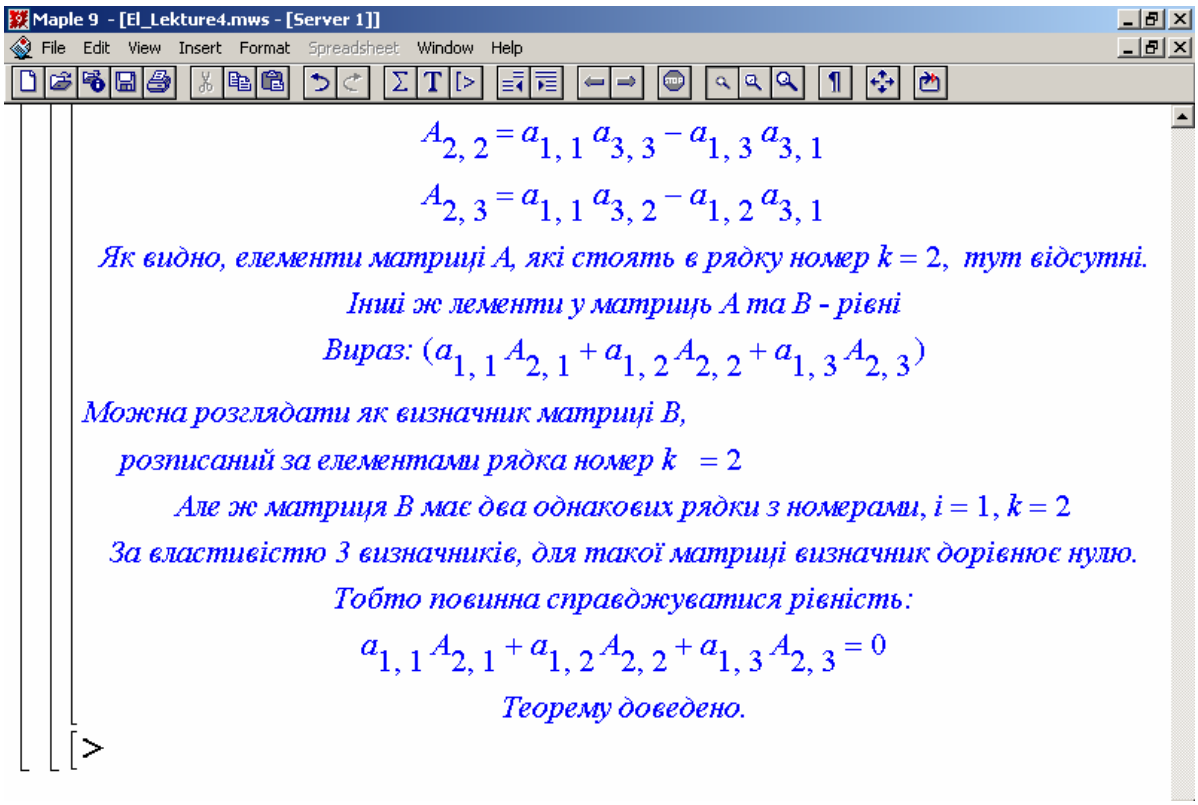
$$a_{1,1}A_{2,1}+a_{1,2}A_{2,2}+a_{1,3}A_{2,3}=0$$

записане для матриць A та B, не змінюється,

оскільки цей вираз не залежить від елементів рядка номер $k=2$

Дійсно:

$$A_{2,1}=a_{1,2}a_{3,3}-a_{1,3}a_{3,2}$$


Maple 9 - [El_Lekture4.mws - [Server 1]]
File Edit View Insert Format Spreadsheet Window Help
$A_{2,2} = a_{1,1}\,a_{3,3} - a_{1,3}\,a_{3,1}$
$A_{2,3} = a_{1,1}\,a_{3,2} - a_{1,2}\,a_{3,1}$
Як видно, елементи матриці A, які стоять в рядку номер $k = 2$, тут відсутні.
Інші ж лементи у матриць A та B - рівні
Вираз: $(a_{1,1}\,A_{2,1} + a_{1,2}\,A_{2,2} + a_{1,3}\,A_{2,3})$
Можна розглядати як визначник матриці B,
розписаний за елементами рядка номер $k = 2$
Але ж матриця B має два однакових рядки з номерами, $i = 1, k = 2$
За властивістю 3 визначників, для такої матриці визначник дорівнює нулю.
Тобто повинна справджуватися рівність:
$a_{1,1}\,A_{2,1} + a_{1,2}\,A_{2,2} + a_{1,3}\,A_{2,3} = 0$
Теорему доведено.
>

Література

1. Манзон Б.М. Maple V Power Edition. - М.: Информационно-издательский дом «Филинъ», 1998. - 240 с.
2. Дьяконов В.П. Математическая система Maple V R3/R4/R5. - М.: Солон, 1998. - 400 с.
3. Говорухин В.Н., Цибулин В.Г. Введение в Maple V. Математический пакет для всех. - М.: Мир, 1997. 207 с.
4. Прохоров Г.В., Леденев М.А., Колбеев В.В. Пакет символьных вычислений Maple V. - М.: Петит, 1997. - 200 с.
5. Матросов А.В. Maple 6. Решение задач высшей математики и механики. – СПб.: БХВ-Петербург, 2001. – 528 с.
6. Дьяконов В.П. Maple 7: учебный курс. - СПб.: Питер, 2002. - 672 с.
7. Васильев А.Н. Maple 8. Самоучитель.: - М.: Издательский дом “Вильямс”, 2003. - 352 с.
8. Волков Ю.І., Найко Д.А. Лінійна алгебра й аналітична геометрія з елементами програмування мовою Паскаль. Навчальний посібник. – К.: НМК ВО, 1990. – 144 с.

Навчальне видання

Володимир Маркусович Михалевич

MAPLE.
КОМП'ЮТЕРНА ПІДТРИМКА КУРСУ ВИЩОЇ МАТЕМАТИКИ В ТЕХНІЧНОМУ ВУЗІ

ЧАСТИНА І.

ЛІНІЙНА Й ВЕКТОРНА АЛГЕБРА. АНАЛІТИЧНА ГЕОМЕТРІЯ

Навчальний посібник

Оригінал-макет підготовлено автором

Редактор В.О. Дружиніна

Навчально-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку Гарнітура Times New Roman
Формат 29,7х42 1/4 Папір офсетний
Друк різографічний Ум. друк. арк.
Тираж прим.
Зам. №

Віддруковано в комп'ютерному інформаційно-видавничому центрі

Вінницького національного технічного університету
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ