Laboratorní cvičení - Lineární algebra

Pro práci na tomto laboratorním cvičení je nejprve potřeba vyvolat balíček programů **LinearAlgebra**určený pro lineární algebru. Matici neoznačujte písmenem D, je použito jako symbol pro derivaci.

Matice

Zápis matice

Zadat matici můžeme několika způsoby, my si ukážeme jen některé z nich. Pro zadávání použijeme příkaz **Matrix.**Více podrobností lze najít v Helpu, např. v examples/LA_Syntax_Shortcuts.

Příklad 1. Napište matici $A = \begin{bmatrix} 1 & 2 \\ -5 & 0 \\ 1 & -28 \end{bmatrix}$.

V tomto případě použijeme zápis tvaru **Matrix(m,n, seznam)**, kde **m**je počet řádků, **n**je počet sloupců dané matice a **seznam**je výčet prvků matice psaný po řádcích.

```
> restart;
> with(LinearAlgebra):
> A:=Matrix(3,2,[1,2,-5,0,1,-28]);
```

$$A := \begin{bmatrix} 1 & 2 \\ -5 & 0 \\ 1 & -28 \end{bmatrix}$$

Příklad 2. Napište matici $B = \begin{bmatrix} 1 & -5 & 2 \\ 2 & 0 & -1 \end{bmatrix}$.

Jiný způsob jak nadefinovat matici je vložit do příkazu Matrix seznam, který obsahuje prvky zapsané po řádcích.

```
> restart;
> with(LinearAlgebra):
> B:=Matrix([[1,-5,2],[2,0,-1]]);
```

$$B := \begin{bmatrix} 1 & -5 & 2 \\ 2 & 0 & -1 \end{bmatrix}$$

Příklad 3. Napište nulovou matici třetího řádu.

```
> C:=Matrix(3,3,0);
```

$$C := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Prvky matice lze definovat jako funkce dvou nezávisle proměnných. Za ně budou dosazeny hodnoty řádkového a sloupcového indexu.Např.

```
> f:=(i,j)->1/x^(i+j);
```

$$f := (i,j) \rightarrow \frac{1}{x^{i+j}}$$

```
> E:=Matrix(4,3,f);
```

$$E := \begin{bmatrix} \frac{1}{x^2} & \frac{1}{x^3} & \frac{1}{x^4} \\ \frac{1}{x^3} & \frac{1}{x^4} & \frac{1}{x^5} \\ \frac{1}{x^4} & \frac{1}{x^5} & \frac{1}{x^6} \\ \frac{1}{x^5} & \frac{1}{x^6} & \frac{1}{x^7} \end{bmatrix}$$

Prvky matice lze definovat libovolným funkčním předpisem se dvěma proměnnými.

```
> g:=(u,v)->u+v-1;
```

$$g := (u, v) \rightarrow u + v - 1$$

```
> X:=Matrix(3,3,g);
```

$$X := \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

Matici lze definovat také z levé lišty pomocí palety ***Matrix.*** V tomto případě můžeme navolit počet řádků a sloupců, typ matice, jakého typu mají být prvky matice,... Poklepem na příkaz **Insert Matrix** se nám pak zobrazí požadovaná matice, do které si můžeme doplnit chybějící prvky.

Příklad. Zadejte diagonální matici, která je typu (2,3). Po zadání požadavků dostaneme:

```
> Matrix([[ m[1,1] , 0 , 0 ],
        [ 0 , m[2,2] , 0 ]], shape=diagonal);
```

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

Fialově vysvícené prvky doplníme dle potřeby.

Příklad. Vytvořte pomocí palety ***Matrix*** symetrickou matici M velikosti 30 s náhodně vygenerovanými prvky.

Všimněte si, že u velkých matic se zobrazí pouze ukazatel s informací o vlastnostech matice. Poklepáním na něj dostanete tabulku, která umožňuje editovat prvky matice.

```
> restart;
> M:=LinearAlgebra:-RandomMatrix(30,30,
outputoptions=[shape=symmetric]);
```

$$M := \begin{bmatrix} \textit{30 x 30 Matrix} \\ \textit{Data Type: anything} \\ \textit{Storage: triangular}_{\textit{upper}} \\ \textit{Order: Fortran\_order} \end{bmatrix}$$

Příklad. Zadejtepomocí palety ***Matrix***jednotkovou matici třetího řádu.

```
> LinearAlgebra:-IdentityMatrix(3,3);
```

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

V dané matici je možné přehazovat řádky a sloupce, vybírat podmatice apod.
Příklad. Z předchozí matice M vyberte podmatici B, tvořenou řádky 5 až 7 a posledními čtyřmi sloupci.

```
> B:=M[5..7,-4..-1];
```

$$B := \begin{bmatrix} -50 & 71 & -55 & -19 \\ -54 & 35 & -46 & 20 \\ 17 & 25 & 19 & -85 \end{bmatrix}$$

Příklad. Z předchozí matice M vyberte podmatici C, tvořenou předposledním řádkem, 20. řádkem a řádky 2 až 5 a 19. až 21. sloupcem, 4. a 9. sloupcem.

```
> C:=M[[-2,20,2..5],[19..21,4,9]];
```

$$C := \begin{bmatrix} 29 & -13 & -89 & 88 & -70 \\ -85 & 58 & -55 & 5 & 92 \\ 24 & -44 & 76 & -2 & -80 \\ 29 & 91 & 41 & -67 & -59 \\ -91 & 5 & -4 & -14 & 30 \\ -38 & -97 & -94 & 45 & -44 \end{bmatrix}$$

Záporné číslo značí řádek nebo sloupec počítaný od konce, kladné číslo značí řádek nebo sloupec počítaný od začátku.

Operace s maticemi

Pro sčítání matic použijeme znaménko "+". Pro násobení matic použijeme znaménko "." (tečka za větou). Pro násobení matice konstantou použijeme znaménko "*" . Pro mocninu matice použijeme znaménko "^". Pro nalezení transponované matice použijeme příkaz **Transpose** nebo "^%T"**.**

Příklad 1. Jsou dány matice $A = \begin{bmatrix} 2 & -5 \\ 6 & -1 \\ 2 & 7 \end{bmatrix}, B = \begin{bmatrix} 1 & -5 & 7 \\ 4 & 2 & 1 \\ -3 & -2 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & -1 & 8 \\ 2 & 5 & -1 \end{bmatrix}, E = \begin{bmatrix} 0 & -6 & 8 \\ 4 & -5 & 9 \end{bmatrix}$. Vypočtěte, pokud to lze, $C+E$, $A \cdot B$, $B \cdot A$, $3 \cdot E$, C^2, B^2, A^T, B^T.
Nejprve si dané matice nadefinujeme.

```
> restart;
> with(LinearAlgebra):
> A:=Matrix(3,2,[2,-5,6,-1,2,7]);
```

$$A := \begin{bmatrix} 2 & -5 \\ 6 & -1 \\ 2 & 7 \end{bmatrix}$$

```
> B:=Matrix(3,3,[1,-5,7,4,2,1,-3,-2,1]);
```

$$B := \begin{bmatrix} 1 & -5 & 7 \\ 4 & 2 & 1 \\ -3 & -2 & 1 \end{bmatrix}$$

```
> C:=Matrix(2,3,[1,-1,8,2,5,-1]);
```

$$C := \begin{bmatrix} 1 & -1 & 8 \\ 2 & 5 & -1 \end{bmatrix}$$

```
> E:=Matrix(2,3,[0,-6,8,4,-5,9]);
```

$$E := \begin{bmatrix} 0 & -6 & 8 \\ 4 & -5 & 9 \end{bmatrix}$$

```
> C+E;
```

$$\begin{bmatrix} 1 & -7 & 16 \\ 6 & 0 & 8 \end{bmatrix}$$

```
> A.B;
Error, (in LinearAlgebra:-Multiply) first matrix column dimension
(2) <> second matrix row dimension (3)
```

V tomto případě dostaneme chybovou hlášku, že matice nejsou stejného typu, tudíž se součin neprovede.

```
> B.A;
```

$$\begin{bmatrix} -14 & 49 \\ 22 & -15 \\ -16 & 24 \end{bmatrix}$$

```
> 3*E;
```

$$\begin{bmatrix} 0 & -18 & 24 \\ 12 & -15 & 27 \end{bmatrix}$$

```
> C^2;
Error, (in rtable/Power) exponentiation operation not defined for
non-square Matrices
```

Pokud není matice čtvercová, daný výpočet se neprovede.

```
> B^2;
```

$$\begin{bmatrix} -40 & -29 & 9 \\ 9 & -18 & 31 \\ -14 & 9 & -22 \end{bmatrix}$$

```
> Transpose(A);
```

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

```
> Transpose(B);
```

$$\begin{bmatrix} 1 & 4 & -3 \\ -5 & 2 & -2 \\ 7 & 1 & 1 \end{bmatrix}$$

```
> B^%T;
```

$$\begin{bmatrix} 1 & 4 & -3 \\ -5 & 2 & -2 \\ 7 & 1 & 1 \end{bmatrix}$$

POZOR! Násobení matic není komutativní.

Příklad 2. Jsou dány matice $A = \begin{bmatrix} 1 & 2 & 0 \\ -4 & 3 & -1 \\ 1 & 4 & 2 \end{bmatrix}, B = \begin{bmatrix} 1 & -12 & 2 \\ 3 & 1 & -2 \\ -3 & 1 & -1 \end{bmatrix}$. Vypočtěte součiny AB a BA.

```
> restart;
> with(LinearAlgebra):
> A:=Matrix(3,3,[1,2,0,-4,3,-1,1,4,2]);
```

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

```
> B:=Matrix([[ 1 , -12 , 2 ],
        [ 3 , 1 , -2 ],
        [ -3 , 1 , -1 ]]);
```

$$B := \begin{bmatrix} 1 & -12 & 2 \\ 3 & 1 & -2 \\ -3 & 1 & -1 \end{bmatrix}$$

```
> A.B;
```

$$\begin{bmatrix} 7 & -10 & -2 \\ 8 & 50 & -13 \\ 7 & -6 & -8 \end{bmatrix}$$

```
> B.A;
```

$$\begin{bmatrix} 51 & -26 & 16 \\ -3 & 1 & -5 \\ -8 & -7 & -3 \end{bmatrix}$$

Determinanty

V případě, že daná matice je čvercová, můžeme počítat její determinant. Výpočet provedeme pomocí příkazu **Determinant(matice).**

Příklad 1. Vypočtěte hodnotu determinantu matice $A = \begin{bmatrix} 1 & 7 & 8 \\ -4 & 5 & 3 \\ -8 & -10 & 0 \end{bmatrix}$ a $B = \begin{bmatrix} 1 & 2 & 3 \\ -2 & -4 & -6 \\ 3 & 2 & 1 \end{bmatrix}$.

```
> restart;
> with(LinearAlgebra):
> A:=Matrix(3,3,[1,7,8,-4,5,3,-8,-10,0]);
```

$$A := \begin{bmatrix} 1 & 7 & 8 \\ -4 & 5 & 3 \\ -8 & -10 & 0 \end{bmatrix}$$

```
> Determinant(A);
```

$$502$$

```
> B:=Matrix(3,3,[1,2,3,-2,-4,-6,3,2,1]);
```

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

```
> Determinant(B);
```

$$0$$

V případě, že chceme vypočítat determinant z matice, která není čtvercová, dostaneme chybové hlášení.

```
> C:=Matrix(2,3,[-3,4,1,5,2,6]);
```

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

```
> Determinant(C);
Error, (in LinearAlgebra:-Determinant) invalid input: 
LinearAlgebra:-Determinant expects its 1st argument, A, to be of type
Matrix(square) but received Matrix(2, 3, {(1, 1) = -3, (1, 2) = 4,
(1, 3) = 1, (2, 1) = 5, (2, 2) = 2, (2, 3) = 6})
```

Soustavy lineárních rovnic

Řešení soustav pomocí příkazu "solve" a "LinearSolve"

Pro řešení soustavy lineárních rovnic používáme v Maple příkaz **solve**. Teto příkaz nám však vrací výsledek pouze v případě, že soustava má právě jedno řešení, nebo jich má nekonečně mnoho.

Příklad 1. Najděte řešení soustavy rovnic $2x + y = 0,$

$$y - 2z = 8,$$
$$x - y + z = 1.$$

```
> restart;
> with(LinearAlgebra):
> solve({2*x+y=0,y-2*z=8,x-y+z=1},{x,y,z});
```

$$\left\{x = \frac{5}{2}, y = -5, z = -\frac{13}{2}\right\}$$

Příklad 2. Najděte řešení soustavy rovnic $2x + y = 0,$

$$y - 2z = 8,$$
$$-3x - 2y + z = -4.$$

```
> solve({2*x+y=0,y-2*z=8,-3*x-2*y+z=-4},{x,y,z});
```

$$\{x = -4 - z, y = 8 + 2z, z = z\}$$

Řešení závisí na jednom parametru, Maple si ho sám zvolí.

Příklad 3. Najděte řešení soustavy rovnic $2x - y - z + 3t = 1,$

$$2x - y - 2t = 4,$$
$$8x - 4y + z - 13t = 19,$$
$$6x - 3y - z - t = 9.$$

```
> solve({2*x-y-z+3*t=1,2*x-y-2*t=4,8*x-4*y+z-13*t=19,
6*x-3*y-z-t=9},{x,y,z,t});
```

$$\{t = t, x = x, y = 2x - 4 - 2t, z = 3 + 5t\}$$

Řešení závisí na dvou parametrech, které si Maple zvolí.
Pokud soustava nemá řešení, tak po "provedení" výpočtu nedostaneme žádnou chybovou hlášku, ale ani se nám nezobrazí žádný výsledek.

Příklad 4. Najděte řešení soustavy rovnic $2x + 2y - z = 1,$

$$2x + y = 0,$$
$$y - z = 4.$$

```
> solve({2*x+2*y-z=1,2*x+y=0,y-z=4},{x,y,z});
```

Řešení jsme nezískali, tato soustava řešení nemá.
Pro řešení soustav používáme také příkaz **LinearSolve(matice,pravé_strany)**. V tomto případě je potřeba chápat soustavu rovnic jako maticovou rovnici tvaru $AX = b$, kde A je matice koeficientů dané soustavy, $b$je vektor pravých stran a X je vektor neznámých, který chceme určit.

Příklad 5. Najděte řešení soustavy $x - 2y + z = 2,$

$$x - y - z = 1,$$
$$3x - 5y + z = 5.$$

```
> restart;
> with(LinearAlgebra):
> A:=Matrix(3,3,[1,-2,1,1,-1,-1,3,-5,1]);
```

$$A := \begin{bmatrix} 1 & -2 & 1 \\ 1 & -1 & -1 \\ 3 & -5 & 1 \end{bmatrix}$$

Vektor pravých stran zadáme příkazem **Vector.**Přestože píšeme složky vektoru do řádku, Maple s nimi pracuje jako se sloupcovým vektorem.

```
> b:=Vector([2,1,5]);
```

$$b := \begin{bmatrix} 2 \\ 1 \\ 5 \end{bmatrix}$$

```
> X:=LinearSolve(A,b);
```

$$X := \begin{bmatrix} 3\_t_3 \\ -1 + 2\_t_3 \\ \_t_3 \end{bmatrix}$$

Příklad 6. Najděte řešení soustavy z příkladu 4.

```
> restart;
> with(LinearAlgebra):
> A:=Matrix(3,3,[2,2,-1,2,1,0,0,1,-1]);
```

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

```
> b:=Vector([1,0,4]);
```

$$b := \begin{bmatrix} 1 \\ 0 \\ 4 \end{bmatrix}$$

```
> X:=LinearSolve(A,b);
Error, (in LinearAlgebra:-LinearSolve) inconsistent system
```

V tomto případě se nám objeví chybová hláška, že systém nemá řešení.

Gaussova a Jordanova eliminační metoda

Při hledání řešení pomocí příkazu solve a LinearSolve dostaneme "hotový" výsledek z Maple, aniž bychom viděli jakoukoli úpravu. Při ručních výpočtech používáme Gaussovu eliminační metodu nebo Jordanovu metodu.

Pro výpočet Gaussovou eliminační metodou v Maple lze použít příkaz **GaussianElimination(matice).** Příkaz převede rozšířenou matici soustavy na horní trojúhelníkovou matici. Chceme-li získat řešení, použijeme příkaz **BackwardSubstitute(upravená_matice).** Dostaneme zápis řešení ve tvaru vektoru. Pro výpočet Jordanovou metodou lze použít příkaz **ReducedRowEchelonForm(matice),**tento příkaz převede matici na diagonální s jedničkami na diagonále. Pokud hledáme řešení, opět napíšeme rozšířenou matici soustavy, po její úpravě příkazem **BackwardSubstitute(upravená_matice)** dostaneme řešení.

Příklad 1. Gaussovou eliminační metodou a Jordanovou metodou řešte soustavu

$3x - y + z = 6,$

$2x - 2z + t = 3,$

$5x - y - z + t = 9,$

$x - y + 3z - t = 3.$

Napíšeme si rozšířenou matici soustavy a provedeme výpočet.

```
> restart;
> with(LinearAlgebra):
> A:=
Matrix(4,5,[3,-1,1,0,6,2,0,-2,1,3,5,-1,-1,1,9,1,-1,3,-1,3]);
```

$$A := \begin{bmatrix} 3 & -1 & 1 & 0 & 6 \\ 2 & 0 & -2 & 1 & 3 \\ 5 & -1 & -1 & 1 & 9 \\ 1 & -1 & 3 & -1 & 3 \end{bmatrix}$$

```
> B1:=GaussianElimination(A);
```

$$B1 := \begin{bmatrix} 3 & -1 & 1 & 0 & 6 \\ 0 & \frac{2}{3} & -\frac{8}{3} & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> X:=BackwardSubstitute(B1);
```

$$X := \begin{bmatrix} \frac{3}{2} + \_t_2 - \frac{1}{2}\,\_t_1 \\ -\frac{3}{2} + 4\,\_t_2 - \frac{3}{2}\,\_t_1 \\ \_t_2 \\ \_t_1 \end{bmatrix}$$

```
> B2:=ReducedRowEchelonForm(A);
```

$$B2 := \begin{bmatrix} 1 & 0 & -1 & \frac{1}{2} & \frac{3}{2} \\ 0 & 1 & -4 & \frac{3}{2} & -\frac{3}{2} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Příkazem **BackwardSubstitute(redukovaná_matice)**získáme řešení soustavy.

```
> X:=BackwardSubstitute(B2);
```

$$X := \begin{bmatrix} \frac{3}{2} + \_t0_2 - \frac{1}{2}\,\_t0_1 \\ -\frac{3}{2} + 4\,\_t0_2 - \frac{3}{2}\,\_t0_1 \\ \_t0_2 \\ \_t0_1 \end{bmatrix}$$

Příklad 2. Budeme hledat pomocí Gaussovy eliminační metody řešení příkladu 4 z předchozí kapitoly.

```
> C:=Matrix([[2,2,-1,1],[2,1,0,0],[0,1,-1,4]]);
```

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

```
> GaussianElimination(C);
```

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

Po úpravě na schodovitý tvar vidíme, že rozšířená matice soustavy má větší hodnost, než matice soustavy, a tudíž řešení skutečně neexistuje.

Řešení soustav pomocí inverzní matice

Pomocí matice inverzní řešíme soustavu pouze v případě, že víme, že soustava má mít jediné řešení.Nejprve si ukážeme, jak lze k dané matici nalézt matici inverzní. Připomeňme, že inverzní matice existuje pouze k regulární matici, tj. ke čtvercové matici, jejíž determinant je nenulový. Inverzní matici vyvoláme příkazem **MatrixInverse(jmeno_matice).**

Příklad 1. K matici $A = \begin{bmatrix} 1 & 1 & 9 \\ 2 & 4 & 1 \\ 0 & 5 & -5 \end{bmatrix}$ nalezněte matici inverzní.

```
> restart;
> with(LinearAlgebra):
> A:=Matrix(3,3,[1,1,9,2,4,1,0,5,-5]);
```

$$A := \begin{bmatrix} 1 & 1 & 9 \\ 2 & 4 & 1 \\ 0 & 5 & -5 \end{bmatrix}$$

Příkazem **Rank(jméno_matice)** zjistíme hodnost matice.

```
> Rank(A);
```

$$3$$

```
> Determinant(A);
```

$$75$$

Daná matice je regulární.

```
> B:=MatrixInverse(A);
```

$$B := \begin{bmatrix} -\frac{1}{3} & \frac{2}{3} & -\frac{7}{15} \\ \frac{2}{15} & -\frac{1}{15} & \frac{17}{75} \\ \frac{2}{15} & -\frac{1}{15} & \frac{2}{75} \end{bmatrix}$$

To, že jsou matice A a B navzájem inverzní, ověříme součinem těchto matic. Výsledkem musí být matice jednotková.

```
> A.B;
```

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

Příklad 2. Pomocí inverzní matice řešte soustavu $x + \ y - \ z = -2,$

$$x - 4y + 2z = -1,$$
$$x - \ y + \ z = 0.$$

Zapíšeme matici soustavy a vektor pravých stran.

```
> restart;
> with(LinearAlgebra):
> A:=Matrix(3,3,[1,1,-1,1,-4,2,1,-1,1]);
```

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

```
> b:=Vector([-2,-1,0]);
```

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

Najdeme matici inverzní.

```
> Determinant(A);
```

$$-4$$

```
> A_1:=MatrixInverse(A);
```

$$A\_1 := \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{4} & -\frac{1}{2} & \frac{3}{4} \\ -\frac{3}{4} & -\frac{1}{2} & \frac{5}{4} \end{bmatrix}$$

Maticí inverzní vynásobíme zleva (násobení matic není komutativní) sloupec pravých stran a získáme řešení soustavy.

```
> X:=A_1.b;
```

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

Cramerovo pravidlo

Cramerovo pravidlo lze použít jen v případě, kdy je matice soustavy regulární, tj. je čtvercová a její determinant je nenulový.
Při řešení postupujeme podle definice, tj. tu neznámou, kterou chceme získat, vypočteme jako podíl $x_i = \frac{|A_i|}{|A|}$, kde $|A_i|$ je determinant, který vznikne z matice soustavy *A*náhradou *i*-tého sloupce sloupcem koeficientů pravých stran.
Při použití nástrojů Maplu lze příklady tohoto typu řešit přepsáním *i*-tého sloupce v matici soustavy sloupcem pravých stran.
Příklad 1. Pomocí Cramerova pravidla najděte neznámé x_1 , x_2, x_3 systému:

$$\begin{aligned} x_1 + 3x_2 + 5x_3 + 7x_4 &= 12, \\ 3x_1 + 5x_2 + 7x_3 + x_4 &= 0, \\ 5x_1 + 7x_2 + x_3 + 3x_4 &= 4, \\ 7x_1 + x_2 + 3x_3 + 5x_4 &= 16. \end{aligned}$$

Nejprve najdeme řešení pro neznámé $x_1 \mathrm{a} x_2$ bez substituce, neznámou x_3 pak vypočteme substitucí.

```
> restart;
> with(LinearAlgebra):
> A:=Matrix(4,4,[1,3,5,7,3,5,7,1,5,7,1,3,7,1,3,5]);
```

$$A := \begin{bmatrix} 1 & 3 & 5 & 7 \\ 3 & 5 & 7 & 1 \\ 5 & 7 & 1 & 3 \\ 7 & 1 & 3 & 5 \end{bmatrix}$$

```
> B:=Matrix(4,1,[12,0,4,16]);
```

$$B := \begin{bmatrix} 12 \\ 0 \\ 4 \\ 16 \end{bmatrix}$$

```
> x1:=
Determinant(Matrix(4,4,[12,3,5,7,0,5,7,1,4,7,1,3,16,1,3,5]))/Dete
rminant(A);
```

$$x1 := 1$$

```
> x2:=
Determinant(Matrix(4,4,[1,12,5,7,3,0,7,1,5,4,1,3,7,16,3,5]))/Dete
rminant(A);
```

$$x2 := -1$$

Nyní si matici A uložíme do pomocné matice P3.
Pozor! Nestačí zadat P3:=A. P3 je jen "přezdívka" pro A, takže změna P3 změní i matici A.

```
> P3:=Matrix(A);
```

$$P3 := \begin{bmatrix} 1 & 3 & 5 & 7 \\ 3 & 5 & 7 & 1 \\ 5 & 7 & 1 & 3 \\ 7 & 1 & 3 & 5 \end{bmatrix}$$

Následujícím příkazem v matici P3 zaměníme prvky ve třetím sloupci sloupcem pravých stran, tj. maticí B.

```
> P3[1..4,3]:=B;
```

$$P3_{1..4,\,3} := \begin{bmatrix} 12 \\ 0 \\ 4 \\ 16 \end{bmatrix}$$

Vypíšeme A i P3, abychom se přesvědčili, jaké mají prvky.

```
> A, P3;
```

$$\begin{bmatrix} 1 & 3 & 5 & 7 \\ 3 & 5 & 7 & 1 \\ 5 & 7 & 1 & 3 \\ 7 & 1 & 3 & 5 \end{bmatrix}, \begin{bmatrix} 1 & 3 & 12 & 7 \\ 3 & 5 & 0 & 1 \\ 5 & 7 & 4 & 3 \\ 7 & 1 & 16 & 5 \end{bmatrix}$$

Maple přiřadil prvku P3[1,3]=12, P3[2,3]=0, P3[3,3]=4 a P3[4,3]=16 ostatní prvky v matici P3 zůstaly zachovány. Obdobně bychom mohli postupovat při záměně řádků, jednotlivých prvků nebo celých částí matice. Více viz Help **Matrix and Vector Entry Selection** a **Matrix and Vector Entry Assignment.**

```
> x3:=Determinant(P3)/Determinant(A);
```

$$x3 := 0$$

Vlastní čísla a vlastní vektory matice

Pro výpočet vlastních čísel a vlastních vektorů použijeme následující příkazy:

CharacteristicMatrix(název_matice, λ), tento příkaz nám napíše charakteristickou matici, tj. od prvků na hlavní diagonále odečte číslo λ. Pro nalezení charakteristického polynomu použijeme příkaz **CharacteristicPolynomial(název_matice, λ).**Vlastní čísla najdeme pomocí příkazu **Eigenvalues(název_matice)**a vlastní vektory příslušné vlastním číslům dostaneme použitím **Eigenvectors(název_matice).**

Příklad 1. Určete vlastní čísla a vlastní vektory matice $A = \begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & -1 \\ 2 & -1 & 0 \end{bmatrix}$.

V řešení použijeme všechny příkazy, i když by nám pro nalezení vlastních čísel a vektorů stačil pouze ten poslední.

```
> restart;
> with(LinearAlgebra):
> A:=Matrix(3,[1,-1,1,1,1,-1,2,-1,0]);
```

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

```
> CharacteristicMatrix(A,lambda);
```

$$\begin{bmatrix} \lambda - 1 & 1 & -1 \\ -1 & \lambda - 1 & 1 \\ -2 & 1 & \lambda \end{bmatrix}$$

```
> CharacteristicPolynomial(A,lambda);
```

$$2 + \lambda^3 - 2\lambda^2 - \lambda$$

Kořeny charakteristického polynomu lze najít také pomocí příkazu **solve.**

```
> solve(%);
```

$$1, 2, -1$$

```
> lambda:=Eigenvalues(A);
```

$$\lambda := \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$$

Příkaz **Eigenvectors** nám vypíše jak vlastní čísla, tak vlastní vektory. Vlastní vektory jsou zapsané do matice, vektory příslušné k vlastním číslům čteme **po sloupcích.**

```
> C,V:=Eigenvectors(A);
```

$$C, V := \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & -\frac{1}{5} \\ 0 & 1 & \frac{3}{5} \\ 1 & 1 & 1 \end{bmatrix}$$

Ověříme správnost výsledku např. pro první vlastní číslo. To máme uloženo v C[1], k němu příslušný vlastní vektor je první sloupec matice V, tj. V[1..3,1].

```
> C[1]*V[1..3,1], A.V[1..3,1];
```

$$\begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix}$$

Pro nalezení vlastních vektorů lze použít i **Eigenvectors(název_matice, output = 'list')**. Tento příkaz vypíše vlastní číslo i s jeho násobností a příslušný vlastní vektor. S výhodou se dá využít u vícenásobných vlastních čísel, kdy předchozí příkaz někdy uvede nesprávně nulové sloupce jako vlastní vektory (viz následující příklad).

Příklad 2. Najděte vlastní čísla a vlastní vektory matice $B = \begin{bmatrix} 4 & -1 & 0 \\ 3 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix}$.

```
> restart;
> with(LinearAlgebra):
> B:=Matrix([[4,-1,0],[3,1,-1],[1,0,1]]);
```

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

```
> Eigenvectors(B,output='list');
```

$$\left[\left[2, 3, \left\{\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}\right\}\right]\right]$$

Řešením je trojnásobné vlastní číslo $\lambda = 2$ a k němu příslušný vektor je $v_{1,2,3} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$.

Použití téhož příkazu bez volby output='list' dá:

```
> Eigenvectors(B);
```

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

```
>
```