Algebra

Polynomy, rovnice a jejich soustavy

V této kapitole si ukážeme několik způsobů jak v maplu pracovat s polynomy. Nejdříve se podíváme na řešení rovnic n-tého stupně. V Maplu na řešení rovnic funguje vynikající funkce solve. Budeme řešit jednoduchou kvadratickou rovnici $x^2 + a^2 x + 4 = 0$.

```
> x[1,2] := solve( x^2+a^2*x+4=0 ,x);
```

$$x_{1,2} := -\frac{a^2}{2} + \frac{\sqrt{a^4 - 16}}{2}, -\frac{a^2}{2} - \frac{\sqrt{a^4 - 16}}{2}$$

Tímto způsobem můžeme řešit i celé soustavy rovnice, které nemusí být nutně lineární.

```
> soustava := {y*z+2*alpha*x+beta=0, x*z+2*alpha*y+beta=0,
  x*y+2*alpha*z+beta=0, x^2+y^2+z^2=1, x+y+z=0};
> solve(soustava);
```

$$soustava := \{x + y + z = 0, x^2 + y^2 + z^2 = 1, x\,y + 2\,\alpha\,z + \beta = 0, x\,z + 2\,\alpha\,y + \beta = 0, y\,z + 2\,\alpha\,x + \beta = 0\}$$

```
Warning, solutions may have been lost
```

Maple nám sic neprovedl úplné vyčíslení, ale to snadno vylepšíme pomocí příkazu allvalues, který odporné funkce RootOf snadno vyčíslí:

```
> allvalues(RootOf(6*z^2-1));
```

$$\frac{\sqrt{6}}{6}, -\frac{\sqrt{6}}{6}$$

Solve je univerzální, poradí si i s některými jinými funkcemi než s polynomy, např. s touto jednoduchou středoškolskou rovnicí:

```
> x = solve(2*sin(x)=2-(cos(x))^2,x);
```

$$x = \frac{\pi}{2}$$

Pomocí solve můžeme řešit i nerovnice:

```
> solve(cos(x)>-x^2+1,x);
Warning, solutions may have been lost
```

$$\mathrm{RealRange}(-\infty, \mathrm{Open}(0)), \mathrm{RealRange}(\mathrm{Open}(0), \infty)$$

Samozřejmě nemůžeme od Maplu čekat nějaké zázraky...

```
> solve(x^6+2*x^5+x^3+3*x^2+1=0, x);
```

$$\text{RootOf}(\_Z^6 + 2\,\_Z^5 + \_Z^3 + 3\,\_Z^2 + 1, \mathit{index} = 1),\ \text{RootOf}(\_Z^6 + 2\,\_Z^5 + \_Z^3 + 3\,\_Z^2 + 1, \mathit{index} = 2),\ \text{RootOf}(\_Z^6 + 2\,\_Z^5 + \_Z^3 + 3\,\_Z^2 + 1, \mathit{index} = 3),\ \text{RootOf}(\_Z^6 + 2\,\_Z^5 + \_Z^3 + 3\,\_Z^2 + 1, \mathit{index} = 4),\ \text{RootOf}(\_Z^6 + 2\,\_Z^5 + \_Z^3 + 3\,\_Z^2 + 1, \mathit{index} = 5),\ \text{RootOf}(\_Z^6 + 2\,\_Z^5 + \_Z^3 + 3\,\_Z^2 + 1, \mathit{index} = 6)$$

Ireducibilitu polynomu, můžeme v Maple zjišťovat takto:

```
> irreduc(x^5-5*x+1);
> irreduc(x^2-2*x+1);
```

$$\mathit{true}$$

$$\mathit{false}$$

Galoisova grupa

Maple umí z polynomu nagenerovat Galoisovu grupu pomocí funkce galois. Parametrem této funkce musí být ireducibilní polynom.

```
> galois(x^4+1);
```

"4T2", { "2[x]2", "E(4)" }, "+", 4, { "(1 2)(3 4)", "(1 4)(2 3)" }

Co dodá maple jako výsledek? Na prvním místě (+V4) je jméno Galoisovy¨grupy, na druhém místě (4) je řád grupy tj. počet prvků v grupě a konečně na třetím místě {(1 3)(2 4)...atd.}je soubor jejích generátorů.

Více o grupách viz kapitola Grupy .

Lineární algebra

Základní definice

Nejdříve se podíváme jak se v maple vlastně definují matice a vektory.
Vektor se v Maplu představuje jako jednorozměrné pole. Takhle tedy vypadá třetí vektor kanonické báze 4-rozměrného prostoru:

```
> array(1..4,[0,0,1,0]);
```

$$[0, 0, 1, 0]$$

Vektor můžeme alternativně (a ekvivalentně) zadefinovat pomocí funkce vector z balíčku linalg. Předchozí vektor může vypadat také takhle:

```
> linalg[vector](4,[0,0,1,0]);
```

$$[0, 0, 1, 0]$$

Obdobně matice je představována (překvapivě ;-)) dvourozměrným polem. Takhle zapíšeme jednotkouvou matici stupně 3.

```
> array(1..3,1..3,[[1,0,0],[0,1,0],[0,0,1]]);
```

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

Matici též můžeme nadefinovat ekvivalentní definicí z balíčku linalg:

```
> linalg[matrix](3,3,[[1,0,0],[0,1,0],[0,0,1]]);
```

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

Poznámka: parametry dimenze u vektoru i u matice můžeme vynechat.

Objekty, s kterými budeme pracovat, už máme zadefinované. Teď se podíváme co s nimi Maple umí. Je toho hodně a ukrývá se to všechno v balíčku linalg. Rovnou si ho celý načteme.

```
> with(linalg);
```

[*BlockDiagonal*, *GramSchmidt*, *JordanBlock*, *LUdecomp*, *QRdecomp*, *Wronskian*, *addcol*, *addrow*, *adj*, *adjoint*, *angle*, *augment*, *backsub*, *band*, *basis*, *bezout*, *blockmatrix*, *charmat*, *charpoly*, *cholesky*, *col*, *coldim*, *colspace*, *colspan*, *companion*, *concat*, *cond*, *copyinto*, *crossprod*, *curl*, *definite*, *delcols*, *delrows*, *det*, *diag*, *diverge*, *dotprod*, *eigenvals*, *eigenvalues*, *eigenvectors*, *eigenvects*, *entermatrix*, *equal*, *exponential*, *extend*, *ffgausselim*, *fibonacci*, *forwardsub*, *frobenius*, *gausselim*, *gaussjord*, *geneqns*, *genmatrix*, *grad*, *hadamard*, *hermite*, *hessian*, *hilbert*, *htranspose*, *ihermite*, *indexfunc*, *innerprod*, *intbasis*, *inverse*, *ismith*, *issimilar*, *iszero*, *jacobian*, *jordan*, *kernel*, *laplacian*, *leastsqrs*, *linsolve*, *matadd*, *matrix*, *minor*, *minpoly*, *mulcol*, *mulrow*, *multiply*, *norm*, *normalize*, *nullspace*, *orthog*, *permanent*, *pivot*, *potential*, *randmatrix*, *randvector*, *rank*, *ratform*, *row*, *rowdim*, *rowspace*, *rowspan*, *rref*, *scalarmul*, *singularvals*, *smith*, *stackmatrix*, *submatrix*, *subvector*, *sumbasis*, *swapcol*, *swaprow*, *sylvester*, *toeplitz*, *trace*, *transpose*, *vandermonde*, *vecpotent*, *vectdim*, *vector*, *wronskian*]

Před používáním funkcí z následujících kapitol je nutné zavést balíček linalg.

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

Vektory a vektorové prostory, unitární prostory

Maple umí z lineárně závislých vektorů najít takové, aby tvořily bázi prostoru, který generují (v R^n).

```
> v1 := vector([1,1,1]);
> v2 := vector([1,-1,1]);
> v3 := vector([5,2,7]);
> v4 := vector([1,0,0]);
> v5 := vector([Pi,exp(1),Pi]);
> v6 := vector([Pi,Pi,Pi]);
> basis({v1,v2,v3,v4,v5});
```

$$v1 := [1, 1, 1]$$
$$v2 := [1, -1, 1]$$
$$v3 := [5, 2, 7]$$
$$v4 := [1, 0, 0]$$
$$v5 := [\pi, \mathbf{e}, \pi]$$
$$v6 := [\pi, \pi, \pi]$$
$$\{v1, v2, v3\}$$

Snadno též najedeme bazi direktního součtu dvou vektorových prostorů.

```
> sumbasis({v1,v2},{v5,v1});
```

$$\{v1, v2\}$$

Podobné je to s průnikem prostorů:

```
> intbasis({v1,v2},{v5,v1});
```

$$\{[1, 1, 1], [\pi, \mathbf{e}, \pi]\}$$

V balíčku linalg můžeme vektory měřit podle různých norem (vytvořených ovšem ve smyslu metrických prostorů, ne z teorie unitárních prostorů). Takhle vypadá klasická eukleidovská norma vektoru v5:

```
> norm(v5,2);
```

$$\sqrt{2\,\pi^2 + (\mathbf{e})^2}$$

A tohle je vektorový součin dvou vektorů:

```
> vektorovy_soucin_v1_a_v5 := crossprod(v5,v1);
```

$$vektorovy\_soucin\_v1\_a\_v5 := [\mathbf{e} - \pi, 0, \pi - \mathbf{e}]$$

Není problém zjistit úhel, který vektory svírají:

```
> simplify(angle(v4,v6));
```

$$\arccos\left(\frac{\sqrt{3}}{3}\right)$$

Někdy je důležité vytvořit z obyčejné báze bázi ortogonální. K tomu slouží Gram-Schmidtův ortogonalizační proces:

```
> GramSchmidt([v1,v2,v3]);
```

$$\left[[1, 1, 1], \left[\frac{2}{3}, \frac{-4}{3}, \frac{2}{3}\right], [-1, 0, 1]\right]$$

Často potřebujeme, aby báze byla i ortonormální. Na tohle je v Maplu funkce normalize. Takhle se normalizuje jeden vektor:

```
> normalize(v5);
```

$$\left[\frac{\pi}{\sqrt{2\,\pi^2+(\mathbf{e})^2}}, \frac{\mathbf{e}}{\sqrt{2\,\pi^2+(\mathbf{e})^2}}, \frac{\pi}{\sqrt{2\,\pi^2+(\mathbf{e})^2}}\right]$$

A takhle pomocí příkazu map normalizujeme celou bázi, získanou Gram-Schmidtovým procesem stejně jako výše.

```
> map(normalize,GramSchmidt([v1,v2,v3]));
```

$$\left[\left[\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}\right], \left[\frac{\sqrt{8}\sqrt{3}}{12}, -\frac{\sqrt{8}\sqrt{3}}{6}, \frac{\sqrt{8}\sqrt{3}}{12}\right], \left[-\frac{\sqrt{2}}{2}, 0, \frac{\sqrt{2}}{2}\right]\right]$$

Operace s Maticemi

Na začátek si zadefinujeme pár pěkných matic s kterými budeme počítat.

```
> A := matrix(3,3,[[5,-9,5],[1,-2,1],[2,3,3]]);
> B := matrix(3,3,[[1,-1,2],[0,2,-1],[1,0,1]]);
```

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

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

Maple umí s maticemi všechny možné operace. Vyčíslují se pomocí funkce evalm:

```
> A+B = evalm(A+B);
> A-B = evalm(A-B);
> A^2 = evalm(A^2);
> k*A = evalm(k*A);
> sin(A) = evalm(sin(A));
```

$$A+B=\begin{bmatrix}6 & -10 & 7\\ 1 & 0 & 0\\ 3 & 3 & 4\end{bmatrix}$$

$$A-B=\begin{bmatrix}4 & -8 & 3\\ 1 & -4 & 2\\ 1 & 3 & 2\end{bmatrix}$$

$$A^2=\begin{bmatrix}26 & -12 & 31\\ 5 & -2 & 6\\ 19 & -15 & 22\end{bmatrix}$$

$$k\,A=\begin{bmatrix}5\,k & -9\,k & 5\,k\\ k & -2\,k & k\\ 2\,k & 3\,k & 3\,k\end{bmatrix}$$

$$\sin(A)=\begin{bmatrix}\sin(5) & -\sin(9) & \sin(5)\\ \sin(1) & -\sin(2) & \sin(1)\\ \sin(2) & \sin(3) & \sin(3)\end{bmatrix}$$

Pro maticové násobení používáme speciální operátor &* :

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

$$A\,B=\begin{bmatrix}10 & -23 & 24\\ 2 & -5 & 5\\ 5 & 4 & 4\end{bmatrix}$$

A takhle se matice transponují.

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

$$A^T=\begin{bmatrix}5 & 1 & 2\\ -9 & -2 & 3\\ 5 & 1 & 3\end{bmatrix}$$

Ještě k transponaci : Maple většinou nerozlišuje vektory sloupcové a řádkové, dokonce je někdy tak drzý, že si je sám otáčí (resp. transponuje) viz. příklad na augment .

Ani zjistit řád matice není problém:

```
> rank(A);
```

$$3$$

Snadno spočítáme i matici inverzní:

```
> A^(-1) = inverse(A);
```

$$\frac{1}{A} = \begin{bmatrix} 9 & -42 & -1 \\ 1 & -5 & 0 \\ -7 & 33 & 1 \end{bmatrix}$$

Též determinant není nic těžkého :

```
> detA :=det(k*A);
```

$$detA := -k^3$$

Maple je dokonce schopen poznat, zda je daná matice ortogonální:

```
> orthog(A);
```

$$false$$

Úpravy matic pomocí různých druhů eliminací najdete v následující kapitole.

Lineární rovnice a matice

Máme-li zadány lineární rovnice, můžeme je pomocí Maplu snadno převést na maticový tvar:

```
> rovnice := {2*x+3*y+z=1,x+3*y=0,x+y+z=3};
> K := genmatrix(rovnice,[x,y,z],flag);
```

$$rovnice := \{x + 3y = 0, x + y + z = 3, 2x + 3y + z = 1\}$$

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

Parametr flag určuje, že se do matice dopíše i sloupec pravých stran. Nicméně je praktičtější parametr flag nahradit jménem proměnné do které chceme uložit vektor pravých stran:

```
> K := genmatrix(rovnice,[x,y,z],'l');
> print(l);
```

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

$$[0, 3, 1]$$

Teď napodobíme parametr flag z předchozího a zároveň si uchováme sloupec pravých stran pro pozdější použití. Příkazem adjung skládáme matice (resp. vektory) horizontálně k sobě.

```
> augment(K,l);
```

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

Maple to umí i nazpátek - z matic do rovnic:

```
> geneqns(K,[x,y,z],l);
```

$$\{x + 3y = 0, x + y + z = 3, 2x + 3y + z = 1\}$$

Tohle byly jen takové hračičky, aby se nám lépe počítalo, teď budeme rovnice řešit. Balíček linalg na to obsahuje speciální funkci linsolve. Její výhodou je, že si slušně poradí i v případě nekonečně mnoha řešení a vypíše příslušné vektory, generující prostor všech řešení. Navíc lze jako její parametry zadávat rovnice v maticovém tvaru a ušetřit si práci s vypisováním všech proměnných. Následující rovnici tak napíšu už jen maticově:

```
> A:=
  matrix(4,4,[[1,2,3,-1],[1,-1,1,2],[1,5,5,-4],[1,8,7,-7]]);
> b := vector(4,[0,4,-4,-8]);
> linsolve(A,b);
```

$$A := \begin{bmatrix} 1 & 2 & 3 & -1 \\ 1 & -1 & 1 & 2 \\ 1 & 5 & 5 & -4 \\ 1 & 8 & 7 & -7 \end{bmatrix}$$

$$b := [0, 4, -4, -8]$$

$$\left[\frac{5}{2}\_t_1 + 6 - \frac{7}{2}\_t_2, \_t_1, -\frac{3}{2}\_t_1 - 2 + \frac{3}{2}\_t_2, \_t_2\right]$$

Jak snadno z tohohle dostat standartní tvar řešení tj. součet jednoho řešení a lineárního prostoru řešení homogení rovnice? Nejdříve použijeme zajímavou funkci nullspace, která nám dodá řešení homogení rovnice $A\,x^T = 0$, tedy spočítá Ker f, kde f je homomorfismu daný maticí A.

```
> reseni_homogeni := nullspace(A);
```

$$reseni\_homogeni := \left\{\left[\frac{-7}{2}, 0, \frac{3}{2}, 1\right], \left[\frac{5}{2}, 1, \frac{-3}{2}, 0\right]\right\}$$

Teď už stačí do zjištěného parametrického řešení dosadit jen nuly za $\_t_1$ a $\_t_2$ {respektive jakékoli jiné konstanty, pokud nás to baví počítat) a dostat vektor řešení [6,0,-2,0].

Pomocí funkce linsolve můžeme též hledat matice přechodu, tedy takovou matici X aby pro matice A a B platilo AX = B.

```
> A := matrix(3,3,[[5,-9,5],[1,-2,1],[2,3,3]]);
> B := matrix(3,3,[[1,-1,2],[0,2,-1],[1,0,1]]);
> linsolve(A,B);
```

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

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

$$\begin{bmatrix} 8 & -93 & 59 \\ 1 & -11 & 7 \\ -6 & 73 & -46 \end{bmatrix}$$

Pokud nejsme úplně líní, můžeme si matice upravovat a následně řešit pomocí různých druhů eliminací. Základní eliminace je gaussova. V Maple se provádí takhle:

```
> gausselim(A);
```

$$\begin{bmatrix} 5 & -9 & 5 \\ 0 & \frac{-1}{5} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Gaussovu eliminaci můžeme provádět i v tělesech s celočíselným dělením mod n - Z_n. Dělá se to pomocí ekvivalentního příkazu Gausselim, který dokonce není z balíčku linalg.

```
> Gausselim(A) mod 5;
```

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

Maple umí i další úpravy, jsou to především různé modifikace Gaussovy eliminace, dále se pokouší upravovat i hermitovsky (bohužel podle mě ne moc užitečným způsobem) viz. příslušné helpy na : ffgausselim , gaussjord , ismith , ihermite .

Jak v Maplu obcházet numerické problémy Gaussovy eliminace je popsáno v listu Základy numerické matematiky .

Nakonec se podíváme, jak lze lineární rovnice řešit v tělesech s celočíselným dělením mod n (tj. v tělesech Z_n).
Následující soustavu vyřešíme v Z_5 . Pro snadnější a přehlednější zápis ji napíšeme v maticovém tvaru a teprve potom přepíšeme do rovnic, které vyřešíme pomocí funkce na řešení rovnic v tělesech Z_n msolve.

```
> A :=
  matrix([[4,1,4,5,3,2],[5,0,6,2,0,5],[2,2,3,6,3,3],[4,3,1,2
  ,1,1]]);
> b := vector([5,1,0,1]);
> rovnice := geneqns(A,[x,y,z,t,u,v],b);
> msolve(rovnice, 5);
```

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

$$b := [5, 1, 0, 1]$$

$$rovnice := \{5x + 6z + 2t + 5v = 1, 2x + 2y + 3z + 6t + 3u + 3v = 0,$$
$$4x + y + 4z + 5t + 3u + 2v = 5, 4x + 3y + z + 2t + u + v = 1\}$$

$$\{t = 2\_Z3 + 3, u = 4\_Z2 + \_Z3, v = 2\_Z2 + 3 + 4\_Z3, x = \_Z2, y = 4, z = \_Z3\}$$

Polynomiální matice a Jordanův kanonický tvar

Nejdříve se podíváme na polynomiální matice a jejich kanonický tvar.

```
> Lambda :=
  matrix([[lambda^2,lambda^2-lambda,3*lambda^2],[lambda^2-la
  mbda,3*lambda^2-lambda,lambda^3+4*lambda^2-3*lambda],[lamb
  da^2+lambda,lambda^2+lambda,3*lambda^2+3*lambda]]);
```

$$\Lambda := \begin{bmatrix} \lambda^2 & \lambda^2 - \lambda & 3\lambda^2 \\ \lambda^2 - \lambda & 3\lambda^2 - \lambda & \lambda^3 + 4\lambda^2 - 3\lambda \\ \lambda^2 + \lambda & \lambda^2 + \lambda & 3\lambda^2 + 3\lambda \end{bmatrix}$$

Polynomiální matice se v Maple upravují pomocí funkce Smith, která ekvivalentními úpravami převádí matici na diagonální tvar:

```
> diagonální := smith(Lambda,lambda);
```

$$diagonální := \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda^2 + \lambda & 0 \\ 0 & 0 & \lambda^3 + \lambda^2 \end{bmatrix}$$

Charakteristický polynom jsme mohli též zjistit pomocí determinantu:

```
> charpolynom := (-1)^3*det(Lambda);
> expand(det(diagonální));
```

$$charpolynom := \lambda^6 + 2\lambda^5 + \lambda^4$$

$$\lambda^6 + 2\lambda^5 + \lambda^4$$

Pomocí funkce jordan se u matic nachází jejich Jordanův kanonický tvar. Z dvou

možných definic tohoto tvaru se používá ta, kde jsou případné jedničky v Jordanových buňkách nad diagonálou.

```
> A :=
  matrix([[3,-1,0,0],[1,1,0,0],[3,0,5,-3],[4,-1,3,-1]]);
> JA := jordan(A,T);
```

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

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

Do parametru T se nám napíše transformační matice. Tedy taková matice, aby platilo: $T^{(-1)}\, A\, T = JA$.
V našem případě vypadá takhle:

```
> print(T);
```

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 6 & 1 & 3 & 1 \\ 7 & 0 & 3 & 0 \end{bmatrix}$$

Pokud mají matice stejný Jordanův kanonický tvar jsou podobné. To testuje Maplovská funkce issimilar:

```
> issimilar(A, JA);
> issimilar(A, transpose(A));
> issimilar(A, A&*A);
```

$$true$$

$$true$$

$$false$$

Takhle se počítá stopa matice A:

```
> tr(A) = trace(A);
> tr(JA) = trace(JA);
```

$$\mathrm{tr}(A) = 8$$

$$\mathrm{tr}(JA) = 8$$

Můžeme najít i charakteristický a minimální polynom či ověřit jejich dělitelnost:

```
> charakteristicky := charpoly(A,x);
> minimalni        := minpoly(A,x);
> jejich_delitelnost = divide(charakteristicky,minimalni);
```

$$charakteristicky := x^4 - 8\,x^3 + 24\,x^2 - 32\,x + 16$$

$$minimalni := x^2 - 4\,x + 4$$

$$jejich\_delitelnost = true$$

Z polynomů už máme jednoznačně určené vlastní čísla, zkontrolujeme zda Maplovská funkce eigenvals na jejich hledání opravdu funguje:

```
> vlastni_cisla_A := eigenvals(A);
> koreny_char_polynomu := solve(charakteristicky=0);
```

$$vlastni\_cisla\_A := 2, 2, 2, 2$$

$$koreny\_char\_polynomu := 2, 2, 2, 2$$

K vlastním číslům se hodí vypočítat i vlastní vektory:

```
> eigenvects(A);
```

$$[2, 4, \{[-1, -1, 1, 0], [1, 1, 0, 1]\}]$$

Výsledek je ve tvaru: vlastní číslo, jeho násobnost, {příslušné vlastní vektory}, další vlastní číslo, jeho násobnost atd.
Nyní ale nastupuje nutnost dva zbylé vlastní vektory dopočítat tak aby byly kolmé na ty stávající.

```
> nullspace(matrix([[0, 0, 1, 1],[1, 1, -1, 0]]));
```

$$\{[-1, 0, -1, 1], [-1, 1, 0, 0]\}$$

Vektory můžeme počítat i pomocí numerické funkce Eigenvals, když přidáme navíc jeden parametr.

```
> Eigenvals(A,vektory);
> print(vektory);
```

$$\mathrm{Eigenvals}\left(\begin{bmatrix} 3 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 3 & 0 & 5 & -3 \\ 4 & -1 & 3 & -1 \end{bmatrix}, vektory\right)$$

$$vektory$$

Jak asi vypadá charakteristická matice (tj. matice X = $\lambda\,E - A$ } a jaký má vztah k charakteristickému a minimálnímu polynomu (u matic s lichým stupněm pozor na znaménka)?

```
> lambdaA := charmat(A,lambda);
> upravena := smith(lambdaA,lambda);
```

```
> (-1)^(4)*det(lambdaA);
> expand(det(upravena));
```

$$lambdaA := \begin{bmatrix} \lambda - 3 & 1 & 0 & 0 \\ -1 & \lambda - 1 & 0 & 0 \\ -3 & 0 & \lambda - 5 & 3 \\ -4 & 1 & -3 & \lambda + 1 \end{bmatrix}$$

$$upravena := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \lambda^2 - 4\lambda + 4 & 0 \\ 0 & 0 & 0 & \lambda^2 - 4\lambda + 4 \end{bmatrix}$$

$$\lambda^4 - 8\lambda^3 + 24\lambda^2 - 32\lambda + 16$$

$$\lambda^4 - 8\lambda^3 + 24\lambda^2 - 32\lambda + 16$$

Symetrické matice a bilineární formy

Verze Maplu, se kterou jsem měl možnost pracovat neumí provádět symetrické (resp. Hermitovské) úpravy na symetrickou (resp. Hermitovskou) matici tak, aby se zachovala signatura. Tímto způsobem tedy nelze nacházet vyjádření formy v polárním tvaru, příslušnou polární bázi a zjišťovat zda definitnost dané matice (resp. formy). Naštěstí tu ale existuje příkaz definite, který definitnost matice otestuje:

```
> A:=matrix([[5,1,1],[1,3,2],[1,2,3]]);
> definite(A, positive_def);
> definite(A, negative_def);
> definite(A, positive_semidef);
> definite(A, negative_semidef);
```

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

$$true$$

$$false$$

$$true$$

$$false$$

Pro úpravy Hermitovských matic by se nám mohla hodit následují věc - transponování zároveň s komplexním sdružováním:

```
> B := matrix([[1,2+3*i],[2,3+i]]);
> htranspose(B);
```

$$B := \begin{bmatrix} 1 & 2 + 3i \\ 2 & 3 + i \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 2+3\,\overline{i} & 3+\overline{i} \end{bmatrix}$$

Další problémy, které spadají hlavně do výuky Základů numerické matematiky jsou popsány v příslušné kapitole listu Základy numerické matematiky.

Grupy

Maple má funkce, které umí pracovat s grupami, pečlivě ukryté v balíčku group.

```
> with(group);
```

[*DerivedS, LCS, NormalClosure, RandElement, SnConjugates, Sylow, areconjugate, center, centralizer, core, cosets, cosrep, derived, elements, groupmember, grouporder, inter, invperm, isabelian, isnormal, issubgroup, mulperms, normalizer, orbit, parity, permrep, pres, transgroup*]

Nejdříve se podíváme jak se v Maplu zadefinuje permutační grupa. Grupa se zadává jednak stupněm (tj. z kolika prvků se bude permutovat) a pak množinou generátorů. Jednotlivé generátory se zapisují pomocí cyklů.

```
> nase_grupa := permgroup(2,{[[2,1]]});
```

nase_grupa := permgroup(2, { [[2, 1]] })

Jak tahle naše gupa vypadá. Nejdříve si tak trochu odskočíme z balíčku group a podíváme se jak vypadá množina 2-prvkových permutací. Na příkaz permute je obsahem balíčku combinat..

```
> with(combinat,permute);
> permute(2);
```

[*permute*]

[[1, 2], [2, 1]]

Život je plný překvapení ;-).
Teď se s použití příkazu groupmember podívame, co naše malinkatá grupa obsahuje. Cyklus [] značí identitu.

```
> groupmember([[2,1]], nase_grupa);
> groupmember([], nase_grupa);
```

true

true

Ano, skutečně na nagenerogvání dvojprvkové permutační grupy nám stačí jeden generátor, když jej dobře zvolíme.

Teď ještě jednou:

```
> nase_grupa2 := permgroup(2, {[]});
> groupmember([], nase_grupa2);
> groupmember([[2,1]], nase_grupa2);
```

$$nase\_grupa2 := \mathrm{permgroup}(2, \{[\ ]\})$$

$$true$$

$$false$$

Z pouhé identity toho skutečně moc nenagenerujeme ;-), můžeme si skládat třeba do aleluja. Abychom si potvrdily, že jsme skutečně dobře počítali "ověříme" si řád ($\sigma(g)$) neboli počet prvků obou grup..

```
> rad_prvni_grupy = grouporder(nase_grupa);
> rad_druhe_grupy = grouporder(nase_grupa2);
```

$$rad\_prvni\_grupy = 2$$

$$rad\_druhe\_grupy = 1$$

Počítat do dvou není tak těžké ;-). Teď se podíváme na přeci jenom trošku těžší kousky - core hledá jádro podgrupy g vůči nadgrupě G, tj. největší normální podgrupu g v G. Normální podgrupa grupy G je taková podgrupa, že je inariantní vůči všem vnitřním automorfismům grupy G. Funkce core má dva parametry, první parametr je jméno podgrupy grupy napsané jako druhý parametr:

```
> grupa_velka := permgroup (5,{[[5,4]],[[1,2,3,4,5]]});
> grupa_mala  := permgroup (5,{[[1,2,3]],[[3,4,5]]});
> jadro := core(grupa_mala,grupa_velka);
> rad_jadra = grouporder(jadro);
> rad_maly  = grouporder(grupa_mala);
> rad_velky = grouporder(grupa_velka);
> pocet_permutaci_pet_prvku = 5!;
```

$$grupa\_velka := \mathrm{permgroup}(5, \{[[5, 4]], [[1, 2, 3, 4, 5]]\})$$

$$grupa\_mala := \mathrm{permgroup}(5, \{[[1, 2, 3]], [[3, 4, 5]]\})$$

$$jadro := \mathrm{permgroup}(5, \{[[1, 2, 3]], [[3, 4, 5]]\})$$

$$rad\_jadra = 60$$

$$rad\_maly = 60$$

$$rad\_velky = 120$$

$$pocet\_permutaci\_pet\_prvku = 120$$

Z čehož mimo jiné plyne, že grupa_mala je normální v grupě všech permutací o pěti prvcích.

Teď si ukážeme jak vypadá centralizátor grupy podle nějakého jejího prvku.

```
> c1 := centralizer(grupa_mala, [[1,2]]);
> rad_c1 = grouporder(c1);
```

$$c1 := \mathrm{permgroup}(5, \{[[3, 4, 5]], [[1, 2], [4, 5]]\})$$

$$rad\_c1 = 6$$

A takhle vypadá centrum naší malé (a koneckonců i velké) grupy.

```
> centrum_maly := center(grupa_mala);
> rad_centra = grouporder(centrum_maly);
```

$$centrum\_maly := \mathrm{permgroup}(5, \{\ \})$$

$$rad\_centra = 1$$

Pro Maple není problémem ani najít Sylowovu p-podgrupu dané grupy.

```
> Sylowova_2_grupa := Sylow(grupa_mala, 2);
> a_jeji_rad = grouporder(Sylowova_2_grupa);
```

$$Sylowova\_2\_grupa := \mathrm{permgroup}(5, \{[[2, 3], [4, 5]], [[2, 5], [3, 4]]\})$$

$$a\_jeji\_rad = 4$$

Sice každá konečná grupa lze vyjádřit jako podgrupa permutační grupy, ale přeci jen počítat pouze s nimi není úplně to ono. Naštěstí Maple dovoluje nadefinovat i jiné druhy grup. Měl by nás v tom uspokojit

```
> nova_grupa :=
  grelgroup({a,b,c,d},{[a,a,a],[b,b,b],[a,b,1/a,1/b],[c,d,1/c,1
  /d],[a,b,c,d]});
```

$$nova\_grupa := \mathrm{grelgroup}\left(\{a, b, c, d\}, \{[a, a, a], [b, b, b], [a, b, c, d], \left[a, b, \frac{1}{a}, \frac{1}{b}\right], \left[c, d, \frac{1}{c}, \frac{1}{d}\right]\}\right)$$

Prvním parametrem je množina generátorů grupy, druhá je pak množina operací, které dávají dohromady 1-kový prvek grupy (tj. v našem případě aaa=1 & bbb=1 & atd.}, přičemž 1/x značí prvek inverzní k prvku x.

K takto vytvořeným grupám můžeme libovolně generovat, libovolné podgtupy. Ty se generují funkcí subgrel. Prvním parametrem je množina generátorů podgrupy (každý je potřeba rovností pojmenovat), druhým parametrem je jméno grupy, ze které generujeme.

```
> prvni_subgrel:=subgrel({x=[a,b,1/a]}, nova_grupa);
```

$$prvni\_subgrel := \mathrm{subgrel}\left(\{x = \left[a, b, \frac{1}{a}\right]\}, \mathrm{grelgroup}\left(\{a, b, c, d\}, \{[a, a, a], [b, b, b], [a, b, c, d], \left[a, b, \frac{1}{a}, \frac{1}{b}\right], \left[c, d, \frac{1}{c}, \frac{1}{d}\right]\}\right)\right)$$

Z podgrup vytvořených pomocí subgrel můžeme nazpět vytvořit zase grupy, které už se neodvolávají na žádné jiné, prostě tak, že se všechny rovnice generátorů přepočítají.

```
> g := grelgroup({a,b},{[a,a,a],[a,b,1/a]});
> pg := subgrel({x=[a,b]},g);
> pres(pg);
```

$$g := \mathrm{grelgroup}\left(\{a, b\}, \{[a, a, a], \left[a, b, \frac{1}{a}\right]\}\right)$$

$$pg := \mathrm{subgrel}\left(\{x = [a, b]\}, \mathrm{grelgroup}\left(\{a, b\}, \{[a, a, a], \left[a, b, \frac{1}{a}\right]\}\right)\right)$$

$$\mathrm{grelgroup}(\{x\}, \{[x, x, x]\})$$

K podgrupám definovaným pomocí subgrel už můžeme nalézt jejich permutační reprezentaci vzhledem k původní grupě - provádí se to takhle.

```
> mala := grelgroup({x,y},{[x,x,y,x,y,y,y],[y,y,x,y,x,x,x]});
> mensi := subgrel({x=[x]},mala);
> permutacni_reprezentace = permrep(mensi);
```

$$mala := \mathrm{grelgroup}(\{x, y\}, \{[x, x, y, x, y, y, y], [y, y, x, y, x, x, x]\})$$

$$mensi := \mathrm{subgrel}(\{x = [x]\}, \mathrm{grelgroup}(\{x, y\}, \{[x, x, y, x, y, y, y], [y, y, x, y, x, x, x]\}))$$

$$permutacni\_reprezentace = \mathrm{permgroup}(8, \{x = [[2, 3, 7, 4, 8, 6, 5]], y = [[1, 2, 6, 8, 3, 4, 5]]\})$$

Pozor na složité grupy - to pak Maple jen počítá a počítá, žere vesele paměť a výpočtu se stejně nedobere.

Poslední zajímavou funkcí z teorie grup, kterou si ukážeme je convert. Pomocí ní můžeme přepočítat vyjádření pomocí cyklů do permutací a naopak. Druhý parametr může být 'discjcyc' - to znamená převod do vyjádření cykly, nebo 'permlist' - značí převod do permutací.

```
> convert([5,4,3,2,1],'disjcyc');
> convert([[5,4]], 'permlist',5);
```

$$[[1, 5], [2, 4]]$$

$$[1, 2, 3, 5, 4]$$

```
>
```

Tímto jsme samozřejmě stále ještě nevyčerpali celý balík group. Zájemce o podrobnější informace a další funkce tak musím odkázat na help na group .

Použitá literatura:

Jindřich Bečvář : Vektorové prostory I,II,III

Procházka : Algebra