Command Cheat Sheet

Matlab

Tento přehled obsahuje klíčové příkazy pro Control System Toolbox a Symbolic Math Toolbox, zaměřené na analýzu systémů, přenosové funkce, stavový prostor a návrh regulátorů.

---

1. Reprezentace systém? (Modely a diskretizace)

Před prací se systémy je nutné je v MATLABu nadefinovat. Lze použít přenosové funkce, stavový prostor, nebo nuly a póly.

Příkaz	Popis
`tf(num, den)`	Vytvoří spojitou přenosovou funkci z vektorů koeficientů čitatele a jmenovatele.
`zpk(z, p, k)`	Vytvoří model z nul, pólů a zesílení.
`ss(A, B, C, D)`	Vytvoří model ve stavovém prostoru ze stavových matic.
`tf(Gss)`	Převede stavový model `Gss` na přenosovou funkci.
`zpk(Gss)`	Převede systém na tvar rozložený na kořeny (nuly/póly).
`c2d(G, Ts, 'zoh')`	Převede spojitý systém na diskrétní s periodou vzorkování `Ts` (Zero-Order Hold).
`d2c(G_disc)`	Zpětný převod z diskrétního do spojitého času.

Příklad v kódu:

```
% --- Stavový prostor ---
A = [1 2; 3 4]; B = [1; 0]; C = [1 1]; D = 0;
Gss = ss(A, B, C, D);

% --- Přenosová funkce ---
s = tf('s');            % Založení Laplaceovy proměnné jako objektu
```

```
G = 1 / (s^2 + 2*s + 1); % Rychlý zápis přenosu

% --- Převod na diskrétní ---
Gd = c2d(G, 0.1, 'zoh'); % Vzorkovací perioda 0.1s
```

---

2. Analýza systém? a charakteristiky

Klíčové příkazy pro zjištění vlastností systému a jeho chování (v čase i frekvenci).

Příkaz	Popis
`pole(G)`	Vrátí póly systému (vlastní čísla matice A - určují stabilitu).
`zero(G)`	Vrátí nuly systému.
`dcgain(G)`	Zesílení v ustáleném stavu (DC gain). Hodí se pro výpočet trvalé regulační odchylky.
`pzmap(G)`	Vykreslí mapu nul a pólů do komplexní roviny.
`step(G)`	Vykreslí přechodovou charakteristiku (odezva na jednotkový skok).
`stepinfo(G)`	Vypíše detaily odezvy: překmit (Overshoot), dobu náběhu (RiseTime) a ustálení (SettlingTime).
`impulse(G)`	Vykreslí impulzní charakteristiku (Diracův impuls).
`lsim(G, u, t)`	Odezva na libovolný vstupní signál `u` v čase `t`.
`bode(G)`	Vykreslí Bodeho frekvenční charakteristiku (amplituda a fáze).
`nyquist(G)`	Vykreslí Nyquistovu křivku.
`margin(G)`	Vykreslí Bodeho graf a ukáže zásobu stability (Amplitudová a fázová bezpečnost).
`rlocus(G)`	Geometrické místo kořenů (Root Locus) pro návrh regulátoru podle zesílení.

---

3. Algebra blokových schémat (Propojování)

Funkce pro redukci složitých schémat na jeden výsledný přenos.

```
G1 = tf(1, [1 1]);
G2 = tf(2, [1 3]);

G_ser = series(G1, G2);     % Sériové zapojení (G1 * G2)
G_par = parallel(G1, G2);   % Paralelní zapojení (G1 + G2)
G_fb  = feedback(G1, G2);   % Zpětnovazební smyčka (G1 v přímé, G2 ve zpětné větvi)
G_fb_jednotkova = feedback(G1, 1); % Jednotková záporná zpětná vazba (častý případ)
```

---

4. Stavový prostor – Vlastnosti a ?ízení (Pole Placement & LQR)

Testy matic stavového prostoru a metody pro výpočet matice zpětné vazby K (kde řízení $u = -Kx$).

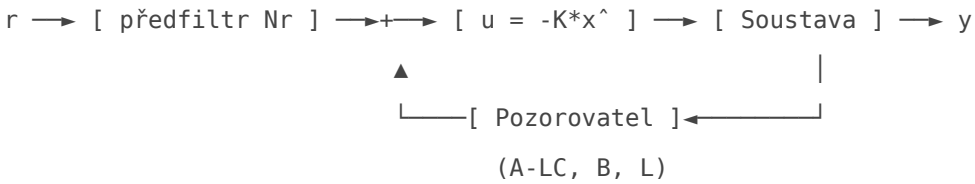
Příkaz	Popis
`Co = ctrb(A, B)`	Matice řiditelnosti (Controllability matrix).
`Ob = obsv(A, C)`	Matice pozorovatelnosti (Observability matrix).
`rank(Co)`	Určí hodnost matice. Pokud `rank(Co) == délka(A)`, systém je plně řiditelný.
`eig(A)`	Vlastní čísla matice A (odpovídá pólům systému).
`K = acker(A, B, p)`	Metoda umístění pólů (pro SISO systémy). Vypočte matici K tak, aby uzavřený systém měl póly `p`.
`K = place(A, B, p)`	Robustnější metoda umístění pólů (i pro MIMO systémy).
`K = lqr(A, B, Q, R)`	Lineární kvadratický regulátor. Najde optimální K na základě penalizačních matic `Q` (stavy) a `R` (akční zásah).

---

5. Stavové ?ízení – Pozorovatel (Luenberger?v observer) a Separa?ní princip

Pokud nejsou všechny stavy měřitelné, je nutné je rekonstruovat pomocí pozorovatele stavu. Výsledný regulátor pak pracuje s odhadnutými stavy $\hat{x}$ místo skutečných x.

Schéma stavového ?ízení s pozorovatelem



Řídící zákon: `u = -K*xˆ + Nr*r` Pozorovatel: `dxˆ/dt = A*xˆ + B*u + L*(y - C*xˆ)`

Výpo?et zisku pozorovatele L

Póly pozorovatele se volí typicky **2–5× rychlejší** (dále vlevo v s-rovině) než póly regulátoru — pozorovatel musí konvergovat dřív, než regulátor začne řídit.

```
% Umístění pólů regulátoru (uzavřená smyčka)
p_reg = [-2+2i, -2-2i, -5];
K = place(A, B, p_reg);

% Póly pozorovatele – zhruba 3x rychlejší
p_obs = 3 * p_reg;   % posunutí dále vlevo
L = place(A', C', p_obs)';  % Dualita: place na transponovaném systému

% Simulace uzavřené smyčky se stavovým regulátorem + pozorovatelem
% Sestavení rozšířeného systému [x; xˆ]:
A_cl = [A - B*K,   B*K;
        zeros(size(A)), A - L*C];
% (kompletní simulaci lze provést přes lsim)
```

P?edfiltr Nr pro nulovou ustálenou odchylku

Bez předfiltru má systém s $u = -Kx$ obecně nenulovou ustálenou odchylku na referenci r.

```
% Výpočet předfiltru:
% Nr = -1 / (C * inv(A - B*K) * B)
Nr = -1 / (C * inv(A - B*K) * B);

% Nebo ekvivalentně přes dcgain uzavřené smyčky:
A_cl = A - B*K;
Nr = 1 / dcgain(ss(A_cl, B, C, 0));
```

Integra?ní složka ve stavovém prostoru (sledování bez odchylky)

Pro robustní sledování reference (eliminace trvalé odchylky i při poruchách) se přidá integrační stav $x_i = \int e , dt$:

```
% Rozšíření soustavy o integrátor chyby
A_aug = [A, zeros(n,1); -C, 0];
B_aug = [B; 0];

% Návrh regulátoru pro rozšířený systém
p_aug = [-2+2i, -2-2i, -5, -1];   % přidán pól pro integrátor
K_aug = place(A_aug, B_aug, p_aug);

% K_aug = [K_x, K_i]  -- stavová zpětná vazba + integrační zisk
```

---

6. Návrh regulátor? – Kam umis?ovat nuly a póly?

Volba polohy pólů a nul regulátoru zásadně ovlivňuje kvalitu odezvy. Níže jsou klíčová pravidla pro nejběžnější typy regulátorů.

Obecná pravidla pro póly uzav?ené smy?ky

Dynamiku uzavřené smyčky popisuje přirozená frekvence ω_n a tlumení ζ dominantního páru pólů:

Požadavek	Kde umístit dominantní póly
Rychlá odezva, malý překmit	$\zeta \approx 0.7$, ω_n co největší (daleko vlevo)
Minimální překmit	$\zeta \geq 1$ (reálné póly, kritické/přetlumení)
Žádná trvalá odchylka	Přidat integrátor (pól v 0) do regulátoru
Stabilita rezerva	Všechny póly s $\text{Re}(s) < 0$, co nejdál od imaginární osy

Nedominantní póly se umísťují **3–5× dál vlevo** než dominantní pár – jejich vliv na odezvu je pak zanedbatelný.

P regulátor

Pouze zesílení – posune póly podél větví kořenového diagramu (rlocus). Nuly ani póly regulátor nepřidává.

```
% Volba zesílení K z rlocus:
rlocus(G);
% Kliknutím na křivku (nebo rlocfind) zjistíme K pro požadované póly:
[K, poles] = rlocfind(G);
```

PD regulátor – nula kompenzuje pomalý pól soustavy

PD regulátor přidává **jednu nulu**: $C_{PD}(s) = K_p(1 + T_d s) = K_p \frac{s + 1/T_d}{1}$

Zlaté pravidlo: Nulu PD regulátoru umísti **na dominantní reálný pól soustavy** (nebo poblíž něj). Tím tento pól „zruší" a soustava se chová rychleji bez výraznějšího překmitu.

```
% Příklad: soustava má pomalý pól v s = -1
G = tf(1, conv([1 1],[1 3]));   % póly v -1 a -3

% Nula PD regulátoru přesně na pól -1 => vykompenzuje ho
Td = 1;                         % 1/Td = 1 => nula v s = -1
C_PD = tf([Td 1], 1);           % C_PD = (s+1)

% Uzavřená smyčka – soustava se chová jako soustava 1. řádu s pólem -3
T = feedback(C_PD * G, 1);
step(T);
```

> ⚠ Přesné zrušení pólu nulou funguje jen v teorii. V praxi zůstane zrušený pól jako „skrytý mód" – systém je stabilní, ale může být citlivý na poruchy nebo nepřesnosti modelu.

PI regulátor – nula blízko po?átku, integra?ní pól v 0

PI regulátor přidává **pól v počátku** (integrátor) a **jednu nulu**: $C_{PI}(s) = K_p \frac{s + 1/T_i}{s}$

Zlaté pravidlo: Nulu PI regulátoru umísti **blízko počátku** (vlevo od nejpomalejšího pólu soustavy, ale blízko něj), aby integrátor výrazně nezpomalil odezvu.

```
% Příklad: soustava s nejpomalejším pólem v s = -0.5
Ti = 2;                          % nula PI v s = -1/Ti = -0.5
C_PI = tf([Ti 1], [Ti 0]);       % C_PI = (s + 0.5)/s

T = feedback(C_PI * G, 1);
step(T);
```

PID regulátor – dv? nuly pro dobrou odezvu

PID regulátor má **pól v počátku** a **dvě nuly**: $C_{PID}(s) = K \frac{(s+z_1)(s+z_2)}{s}$

Zlaté pravidlo: Obě nuly umísti poblíž dominantního páru komplexních pólů soustavy (nebo jako komplexní pár blízko žádaných pólů uzavřené smyčky). Tím regulátor „přitáhne" větve kořenového diagramu do požadované oblasti.

```
% Příklad: dominantní póly soustavy v s = -1 ± 2i
% => Nuly PID umístíme poblíž: např. jako reálný pár -1 ± epsilon
z1 = 1.5; z2 = 0.8;
C_PID = tf(conv([1 z1],[1 z2]), [1 0]);  % (s+1.5)(s+0.8)/s

% Nebo přímo přes pid():
Kp = 2; Ki = 1; Kd = 0.5;
C = pid(Kp, Ki, Kd);

T = feedback(C * G, 1);
stepinfo(T)
```

P?ehled – kam umís?ovat nuly a póly regulátoru

Regulátor	Přidané póly	Přidané nuly	Doporučení pro umístění nul
P	—	—	(žádné nuly)
PD	—	1 nula	Na dominantní reálný pól soustavy (kompenzace)
PI	pól v 0	1 nula	Vlevo od nejpomalejšího pólu soustavy, blízko počátku

Regulátor	Přidané póly	Přidané nuly	Doporučení pro umístění nul
PID	pól v 0	2 nuly	Jako komplexní pár blízko žádaných dominantních pólů
Stavový reg. (place/lqr)	—	—	Přímá volba pólů uzavřené smyčky; nedominantní 3–5× rychlejší

Rychlý postup návrhu pomocí Root Locus

```
G = tf(1, [1 3 2]);     % soustava
C = tf([1 1.5], [1 0]); % regulátor s nulou v -1.5 a pólem v 0 (PI)

rlocus(C * G);           % zobrazí kořenový diagram kompenzované soustavy
% => posouváme nuly/póly C dokud větve neprochází požadovanou oblastí
sgrid(0.7, [])           % přidá iso-křivky tlumení (zeta = 0.7)
```

---

7. Návrh PID regulátor?

Příkazy pro rychlou definici a automatické ladění PID regulátorů.

```
% Definice vlastního PID regulátoru
Kp = 1; Ki = 0.5; Kd = 0.1;
C_pid = pid(Kp, Ki, Kd);

% Automatické ladění na základě přenosu soustavy G
C_tuned = pidtune(G, 'PID'); % Navrhne optimální PID
C_pi = pidtune(G, 'PI');     % Navrhne pouze PI regulátor

% Připojení regulátoru k soustavě (zpětná vazba)
T_uzavrena_smycka = feedback(C_tuned * G, 1);
step(T_uzavrena_smycka);     % Otestování výsledné odezvy
```

---

8. Symbolic Math Toolbox a Linearizace

Skvělé pro analytické řešení diferenciálních rovnic, výpočty přechodových matic a linearizaci nelineárních modelů.

Analytické řešení a stavová přechodová matice:

```
syms s t          % Založení symbolických proměnných
A = [1 2; 0 3];
x0 = [1; 0];      % Počáteční podmínky

% Matice přechodu pomocí exponenciály:
Phi_t = expm(A*t);  % !!! expm() je maticová exponenciála, NEPLÉST s exp() !!!
pretty(Phi_t);      % Graficky hezčí výpis v příkazové řádce

% Řešení přes Laplaceovu transformaci:
I = eye(2);         % Jednotková matice 2x2
X_s = inv(s * I - A) * x0; % (sI - A)^-1 * x0
x_t = ilaplace(X_s);       % Inverzní Laplace - návrat do časové oblasti
```

Linearizace pomocí Jacobiho matice:

```
syms x1 x2 u
% Nelineární diferenciální rovnice: dx/dt = f(x,u)
f1 = -x1^2 + x2 + u;
f2 = x1 - x2*u;

% Linearizace (nalezení matic A, B)
A_sym = jacobian([f1; f2], [x1, x2]); % Derivace podle stavů (A)
B_sym = jacobian([f1; f2], u);        % Derivace podle vstupů (B)

% Pro zjištění číselné matice dosaď pracovní bod (např. x1=0, x2=0, u=1):
A_prac_bod = subs(A_sym, [x1, x2, u], [0, 0, 1]);
B_prac_bod = subs(B_sym, [x1, x2, u], [0, 0, 1]);
```

Revision #2
Created 2026-03-14 10:32:52 UTC by Jakub Kecek
Updated 2026-05-20 07:07:30 UTC by Jakub Kecek