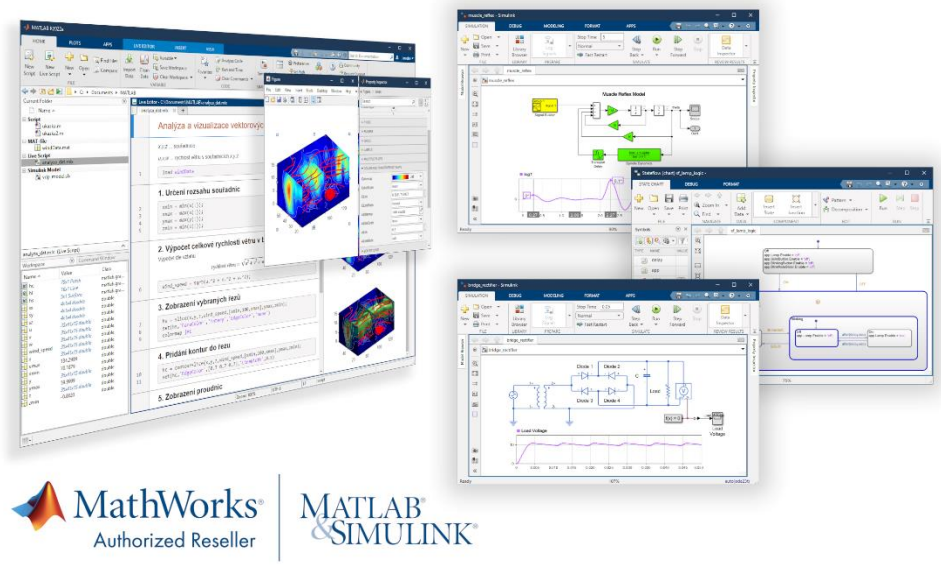
HUMUSOFT®

AI v návrhu řídicích systémů



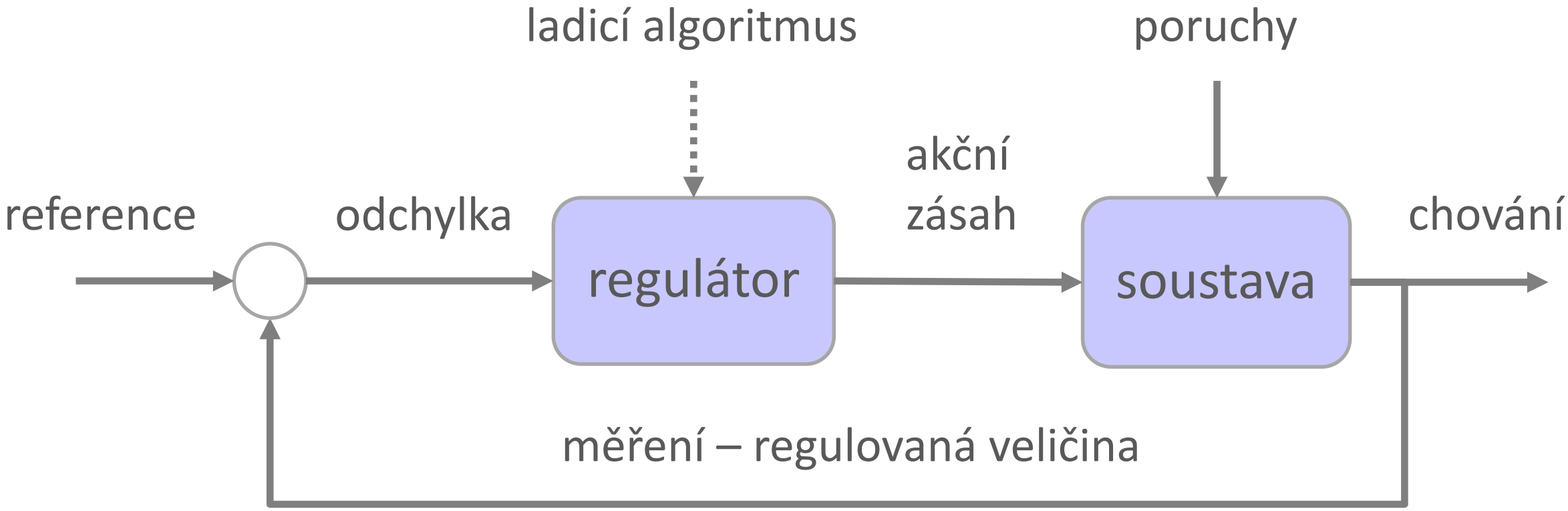
Jaroslav Jirkovský

jirkovsky@humusoft.cz

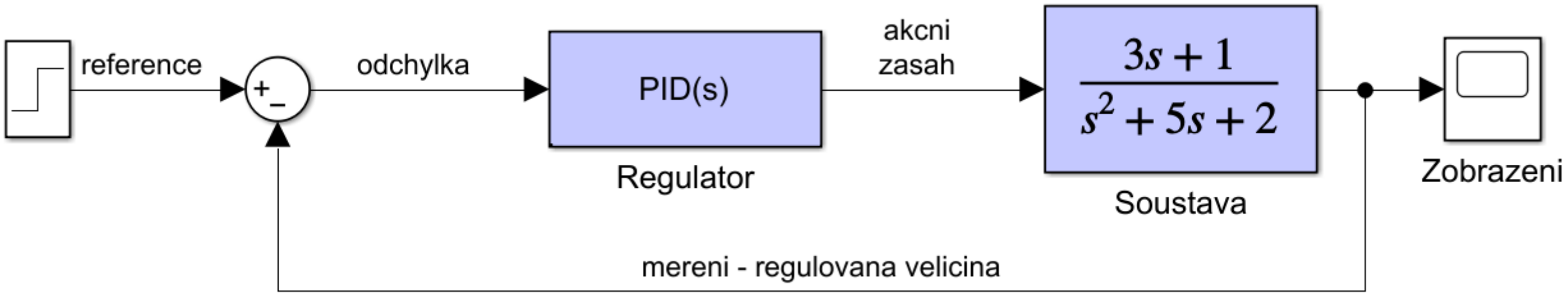
www.humusoft.cz
info@humusoft.cz

www.mathworks.com

Základní schéma řídicího systému

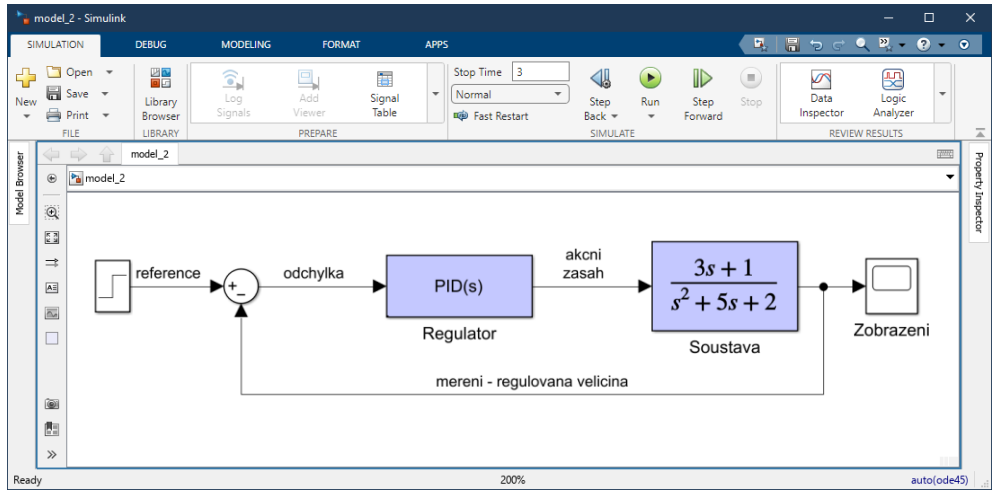


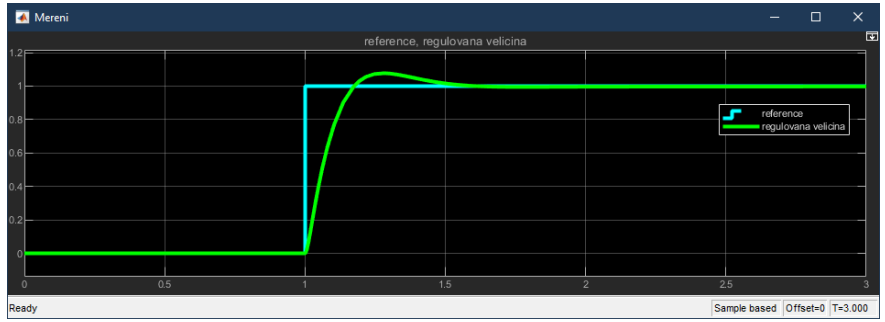
Základní schéma řídicího systému – Simulink



Návrh řídicích systémů v prostředí MATLAB a Simulink

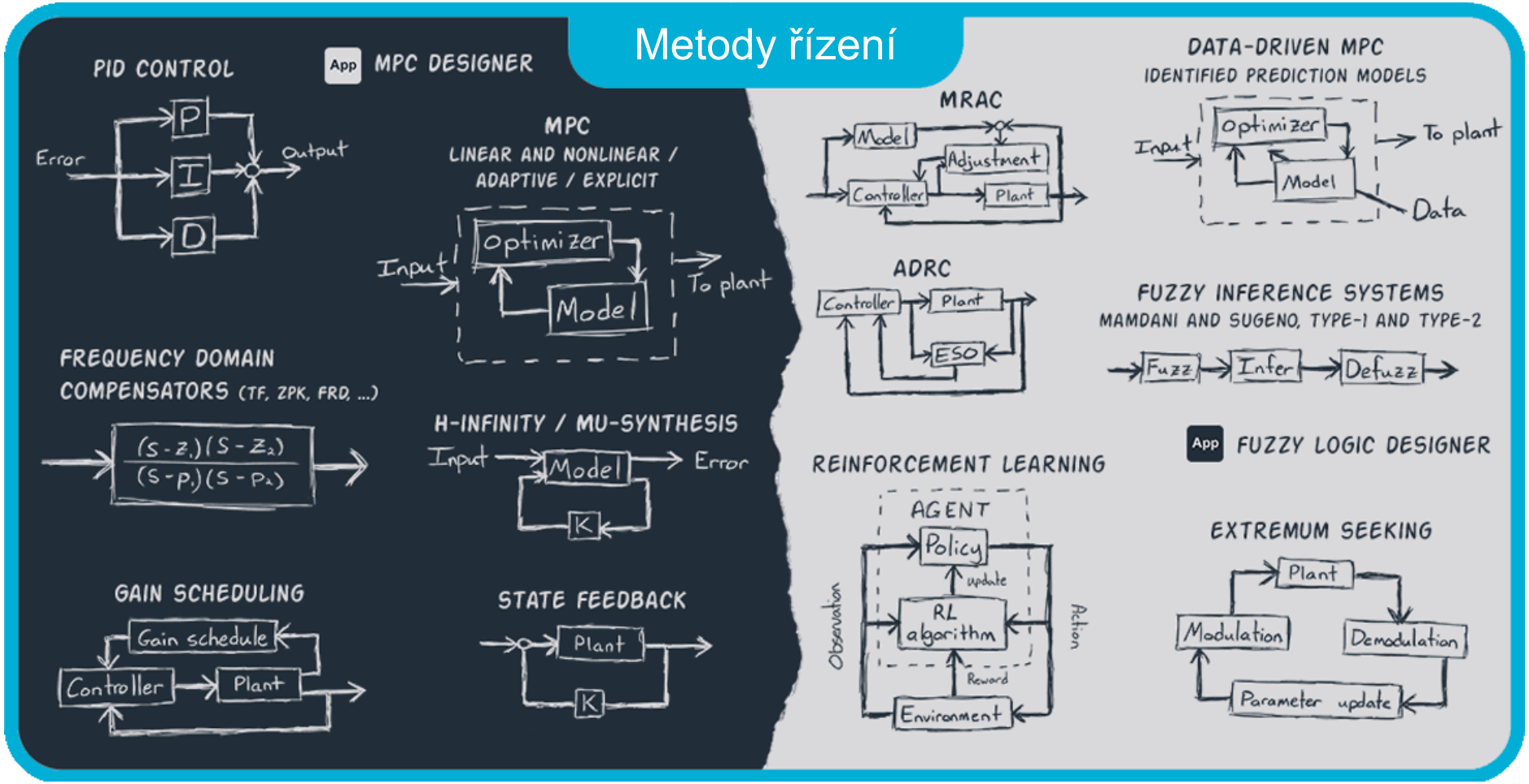
- Propojení modelů soustav s regulátory
  - spojité a diskrétní prvky v jednom modelu
  - bohaté knihovny vstupních signálů
- Libovolná architektura řídicích systému
  - blok PID regulátoru v mnoha variantách
  - spojité, diskrétní, stavové regulátory
  - adaptivní a prediktivní řízení
- Nástroje pro ladění řídicích systémů





Návrh řídicích systémů v prostředí MATLAB a Simulink

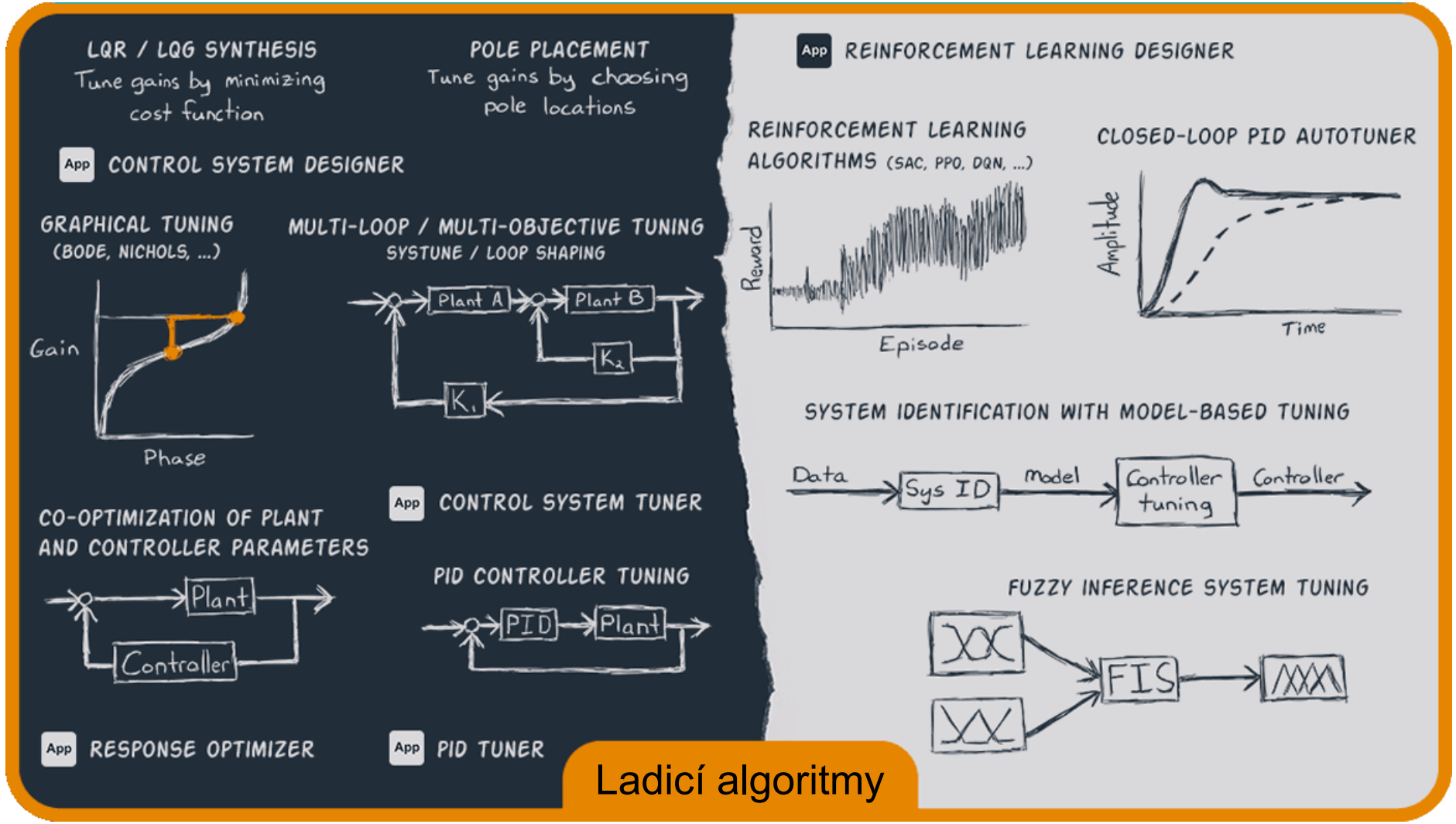
Tradiční



AI a datově orientované

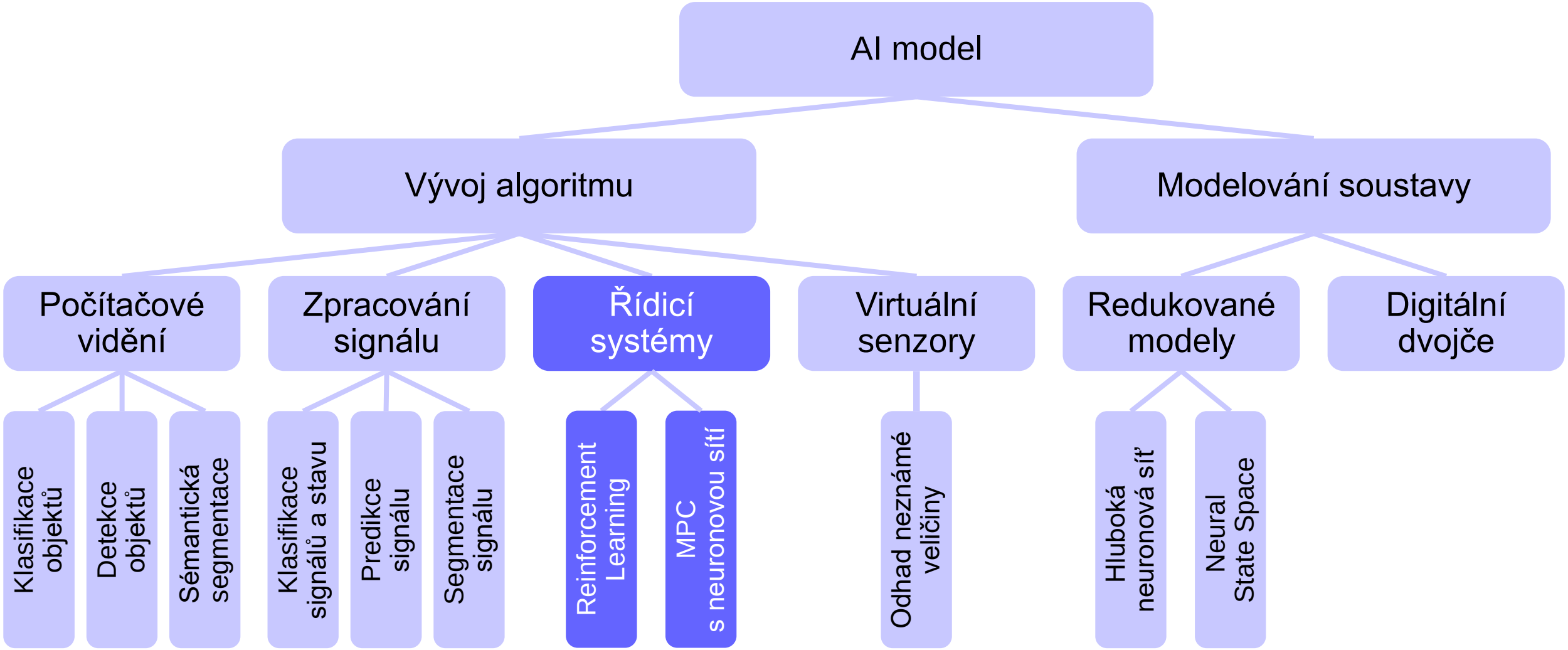
Návrh řídicích systémů v prostředí MATLAB a Simulink

Tradiční



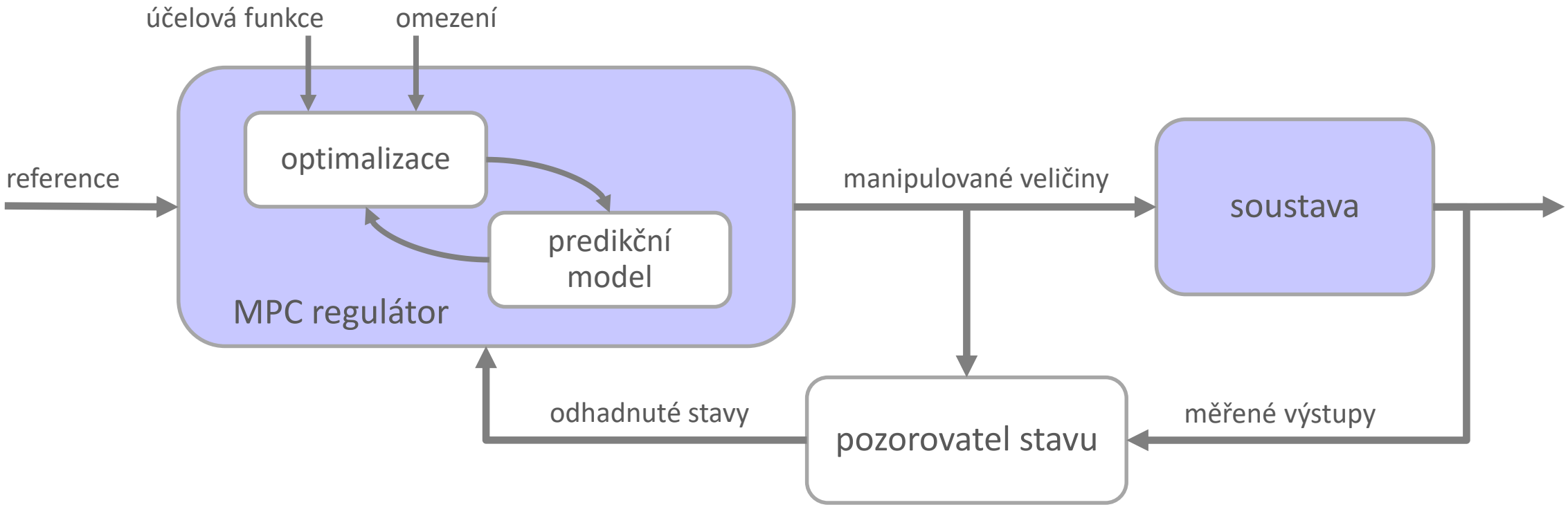
AI a datově orientované

Modelování AI – Typ algoritmu



Model Predictive Control (MPC)

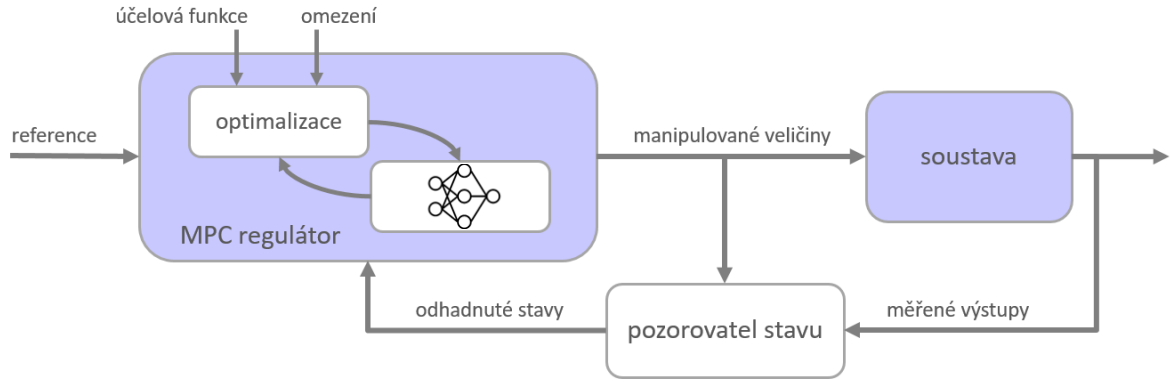
- Prediktivní řízení založené na modelu systému a optimalizaci v reálném čase
  - počítá optimální akční zásah podřízený zadaným omezením



https://www.mathworks.com/videos/series/understanding-model-predictive-control.html

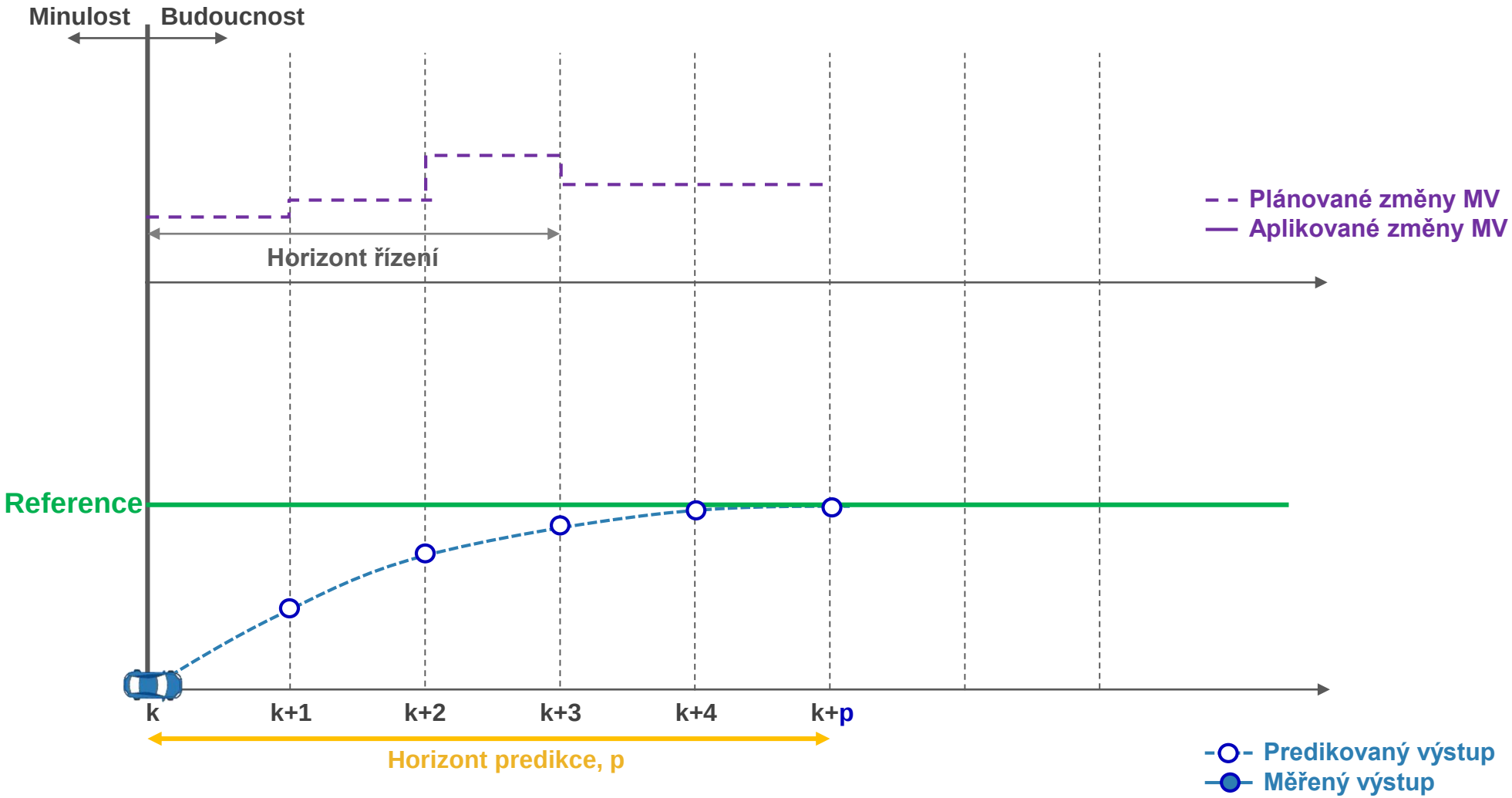
Model Predictive Control (MPC)

- Pro predikci využívá vnitřní model(y)
  - lineární
  - nelineární
- MPC + AI
  - predikční model může být ve formě neural state space modelu
- Typické využití MPC
  - řízení technologických procesů
  - autonomní řízení vozidel
  - robotika



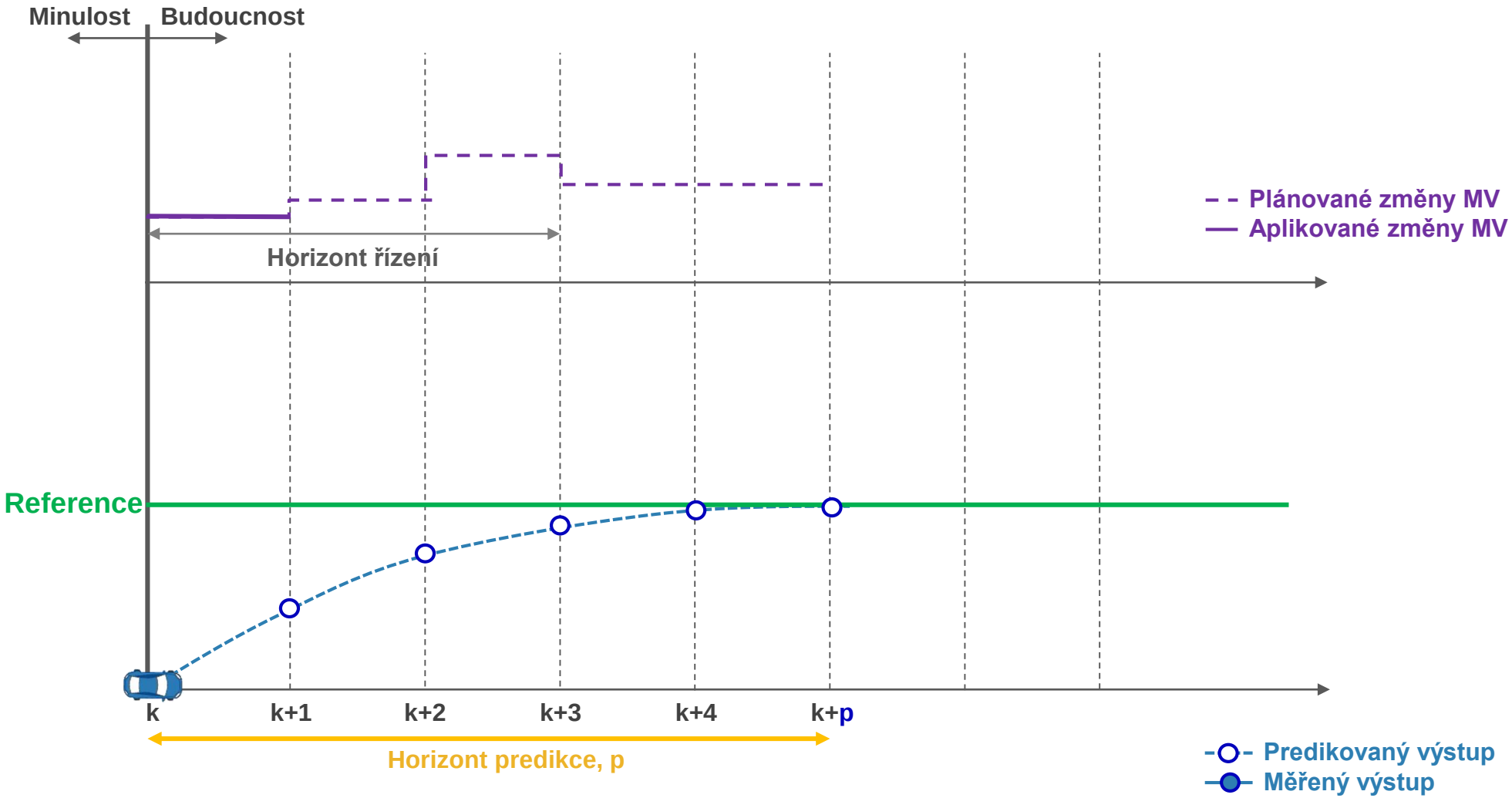
Jak MPC funguje

- Řeší optimalizační úlohu v kroku k, jaký optimální akční zásah provést k dosažení cíle



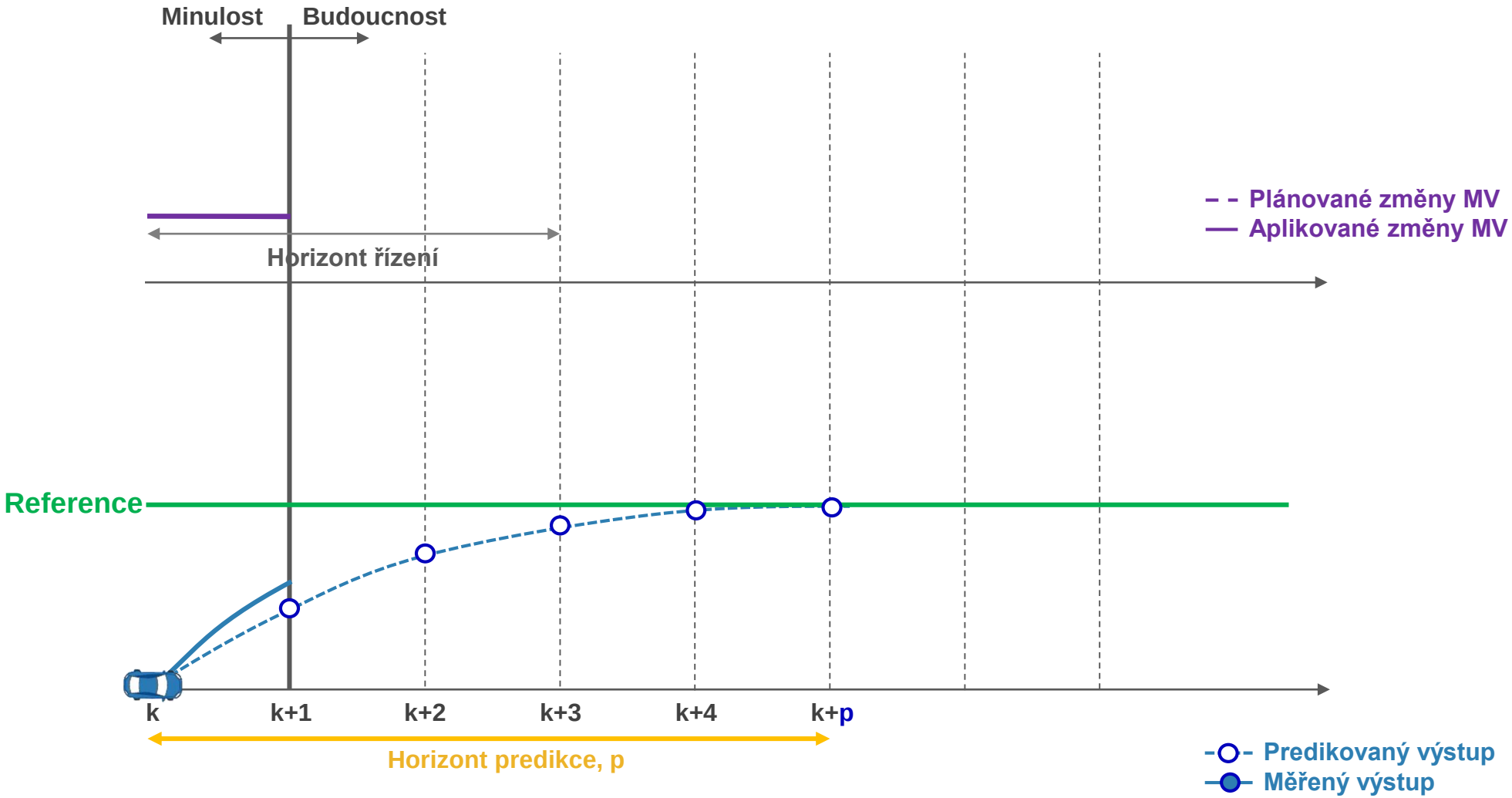
Jak MPC funguje

- Provede první krok řízení (otevřená smyčka), vyřadí zbytek



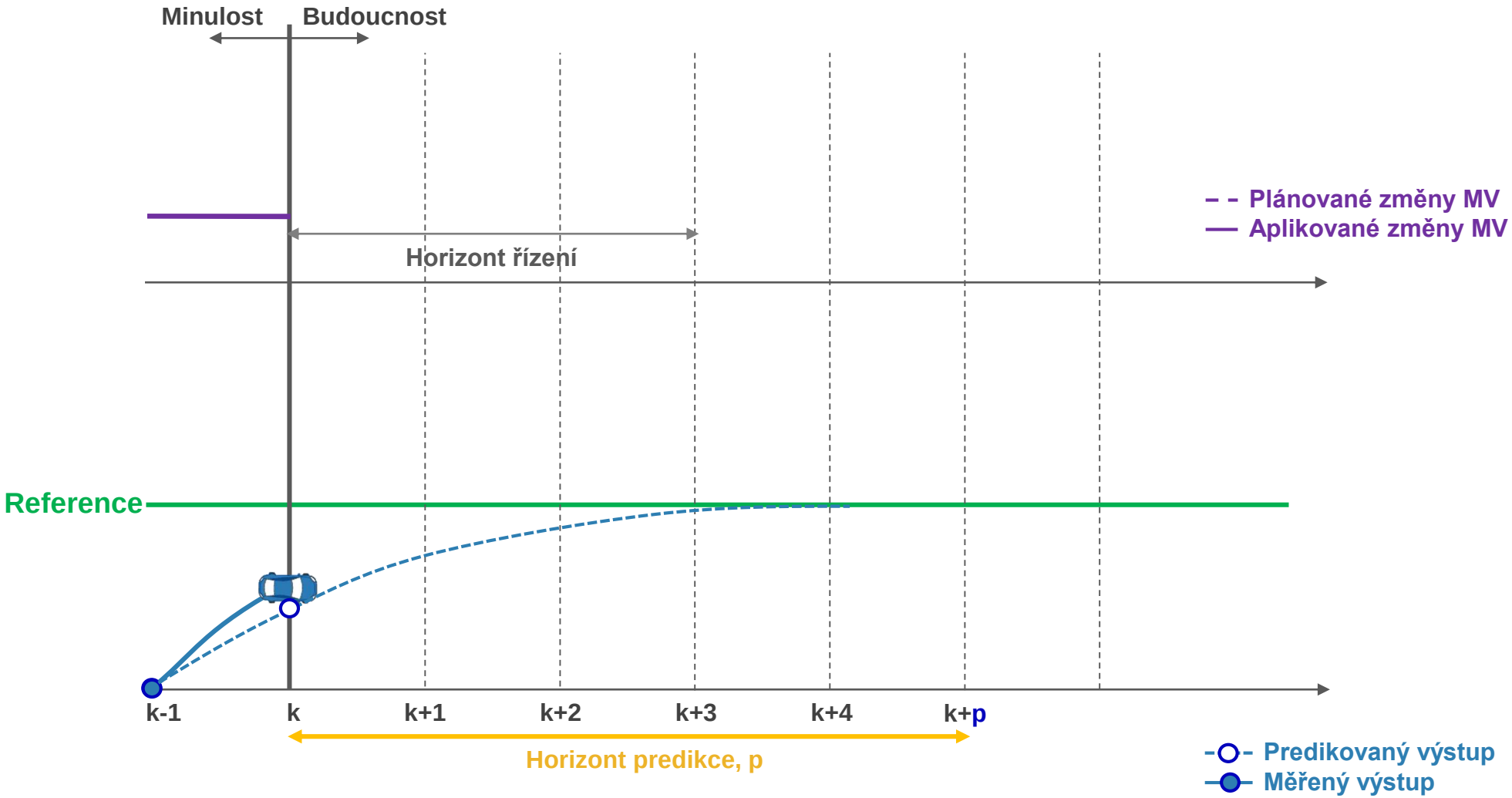
Jak MPC funguje

- Počká na reakci systému



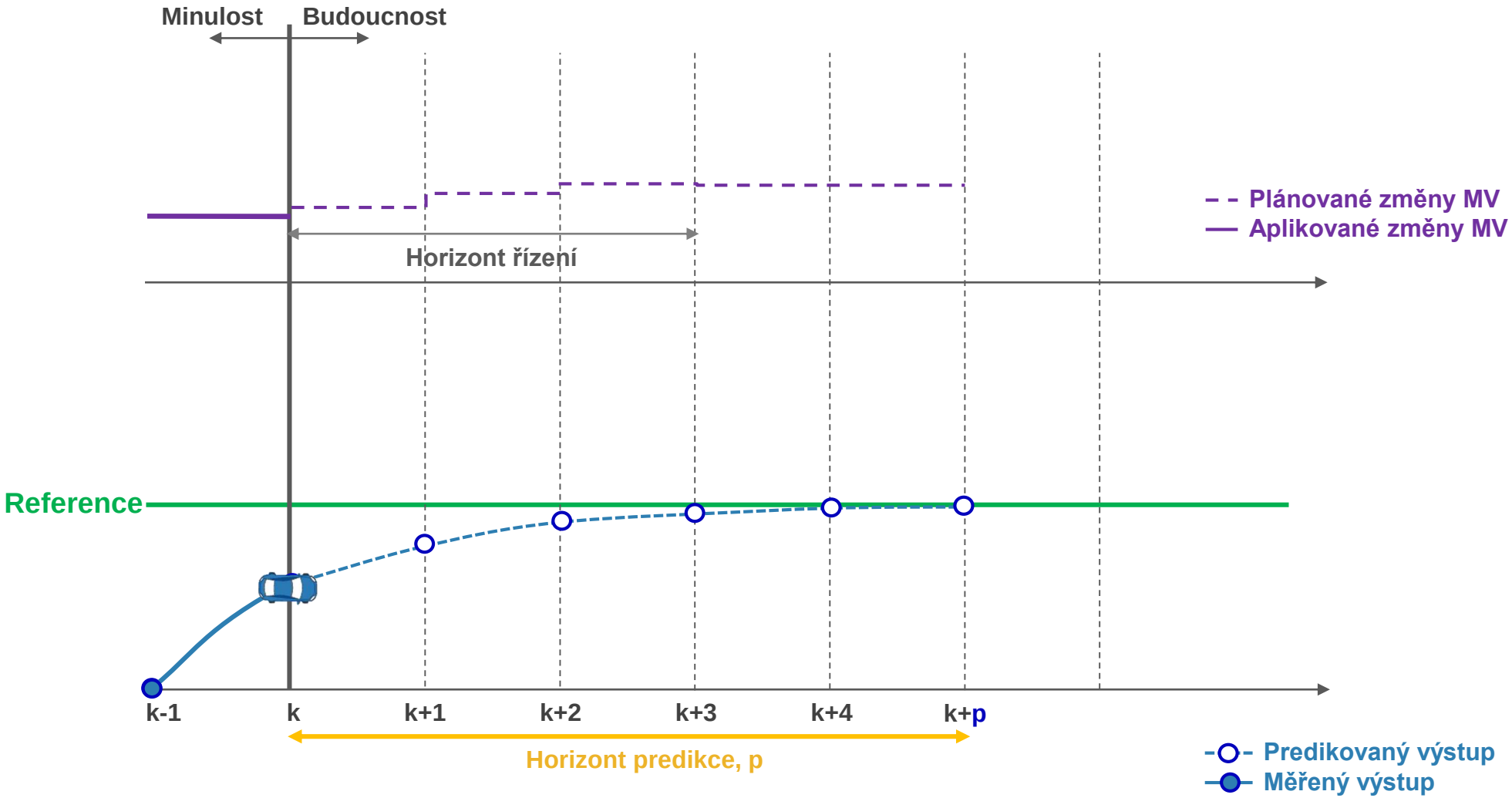
Jak MPC funguje

- Posune časový horizont, změří aktualizované výstupy



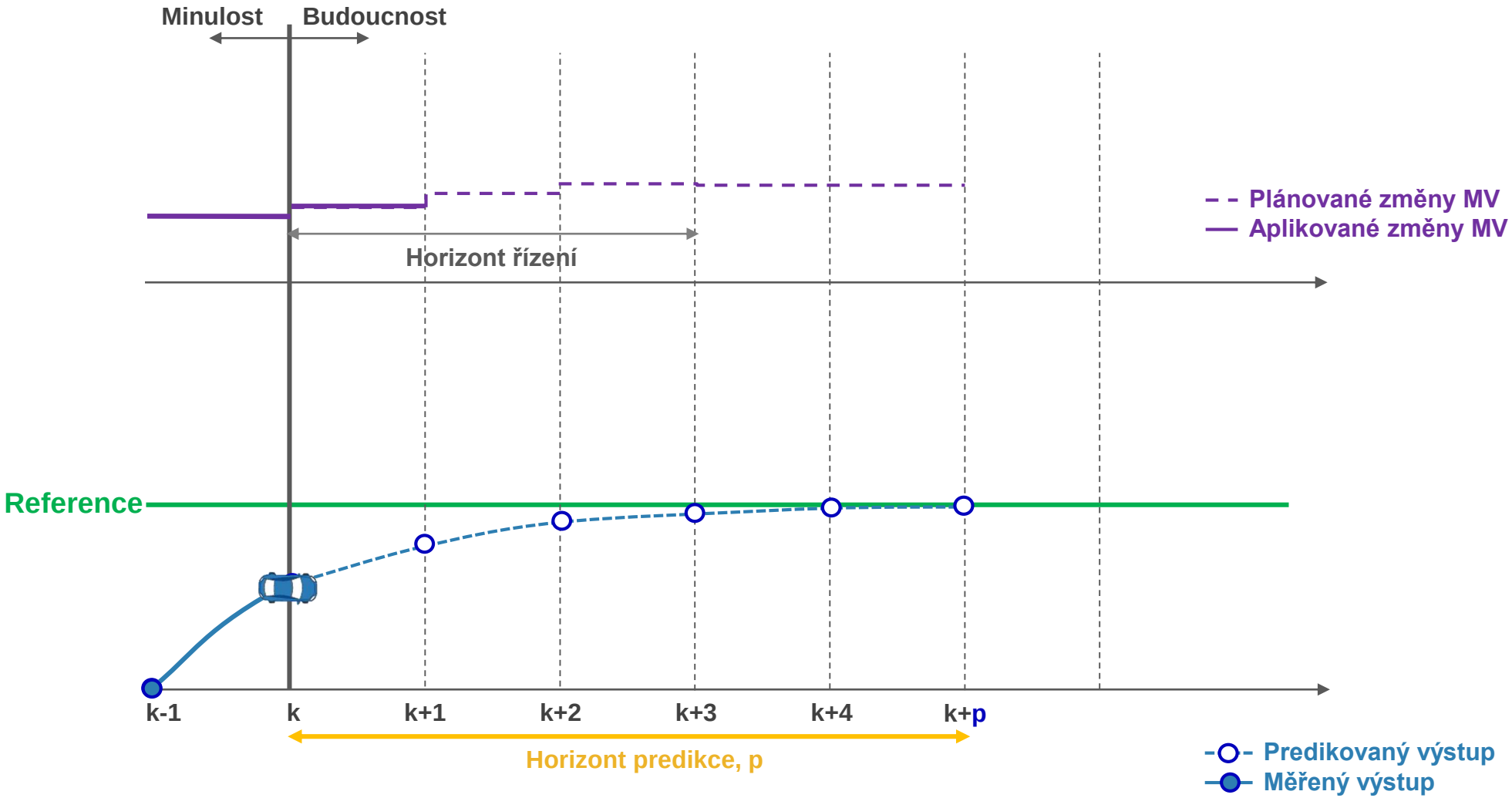
Jak MPC funguje

- Řeší optimalizační úlohu v kroku k



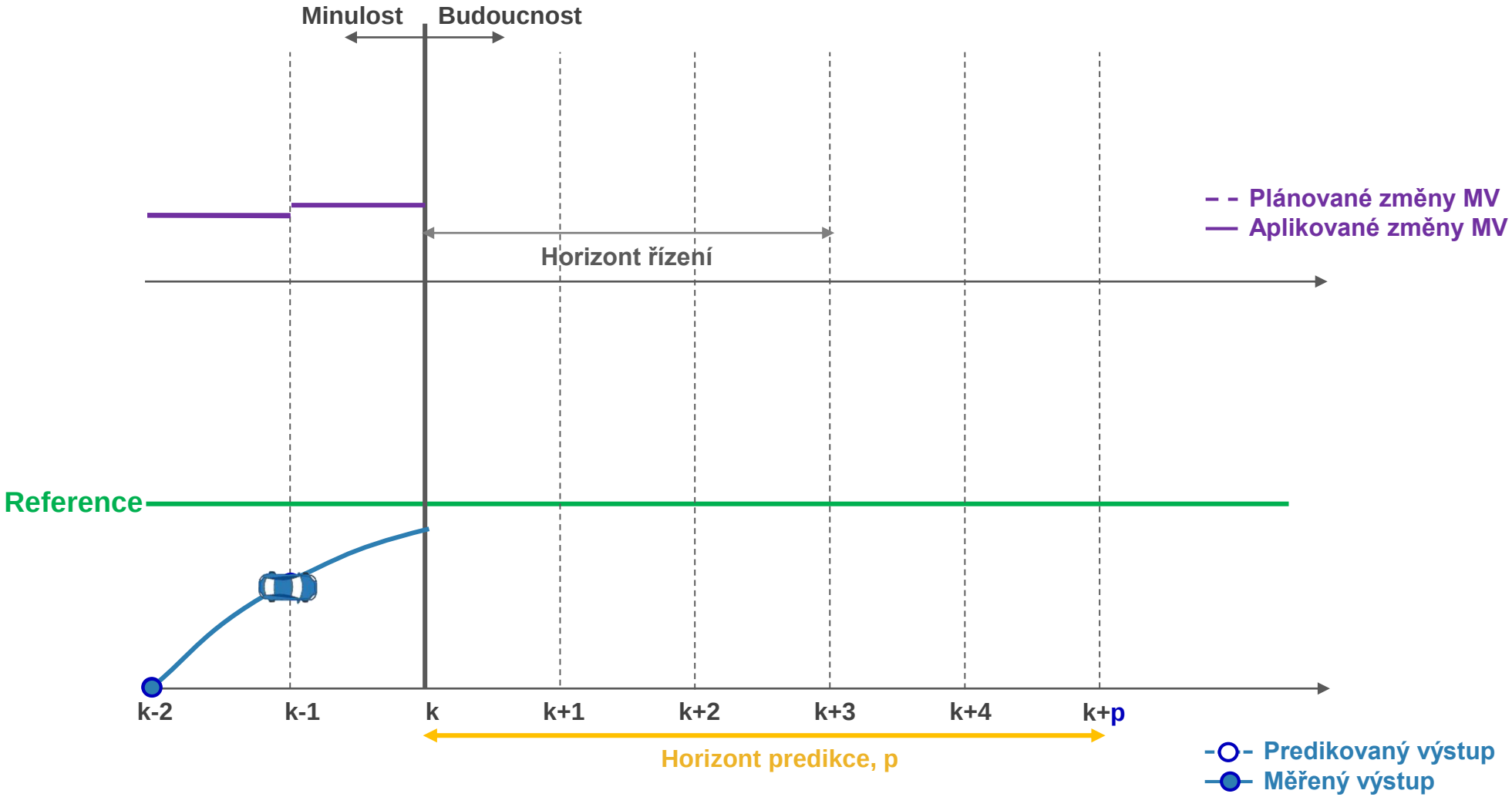
Jak MPC funguje

- Provede první krok řízení (otevřená smyčka), vyřadí zbytek



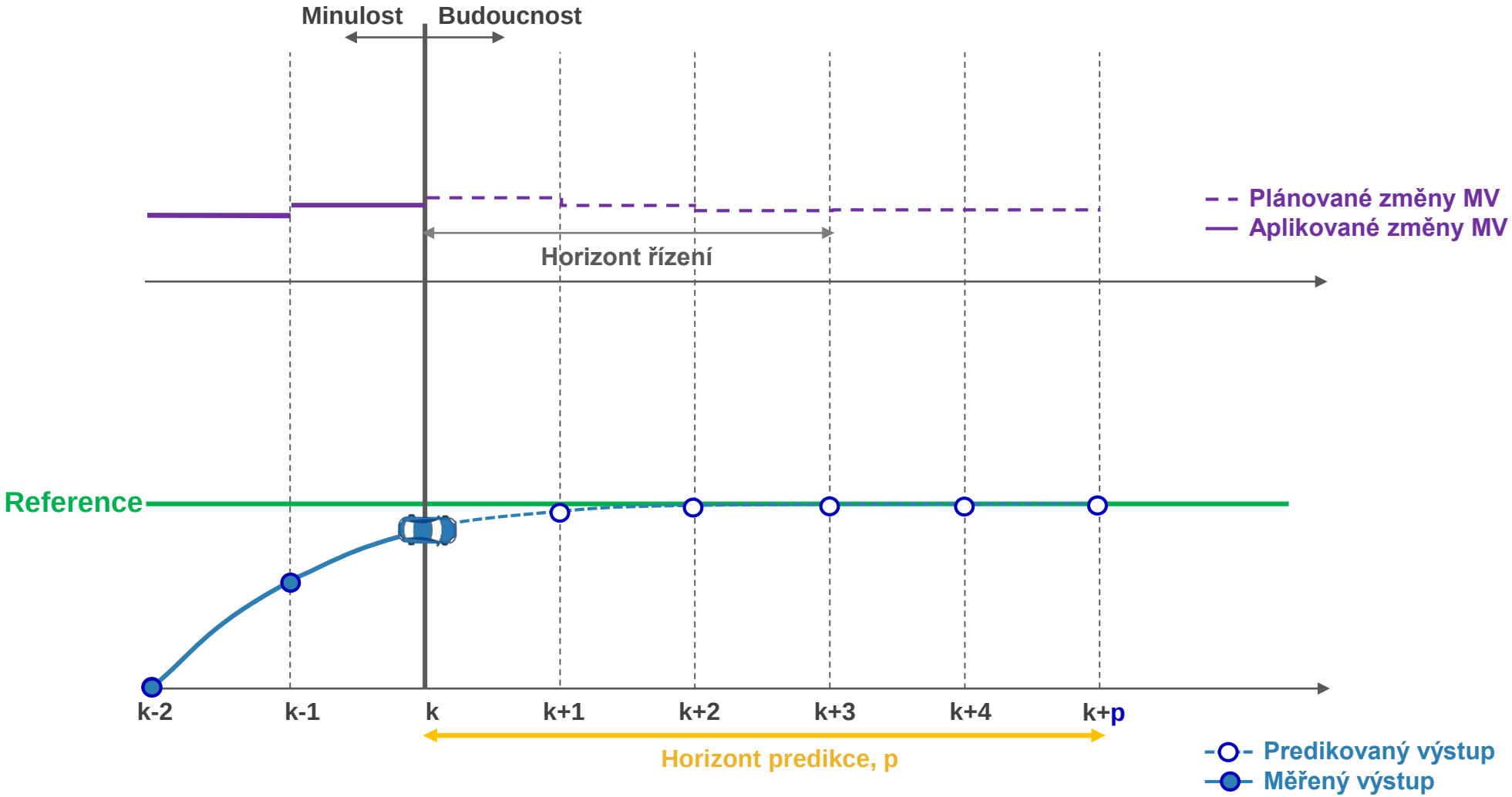
Jak MPC funguje

- Počká na reakci systému, posune časový horizont, změří aktualizované výstupy



Jak MPC funguje

- Řeší optimalizační úlohu v kroku k



Porovnání MPC s tradičními metodami

- Výhody
  - multivariable controller
    - ovládá všechny akční (manipulované) veličiny zároveň
    - umí se vypořádat se vzájemnou provázaností účinků akčních veličin na soustavu
  - zahrnuje omezení
    - počítá optimální akční zásah *podřízený zadaným omezením*
  - prediktivní schopnost
    - počítá dopředu, co může nastat
    - rozhoduje se včas na základě predikce
- Nevýhody
  - výpočetně náročné, náročné na paměť
  - obtížné posouzení standardními metrikami (bezpečnost v amplitudě a fázi, ...)
  - musíme mít dobrý predikční model

Neural State-Space model

- Nelineární MPC může využívat Neural State-Space model

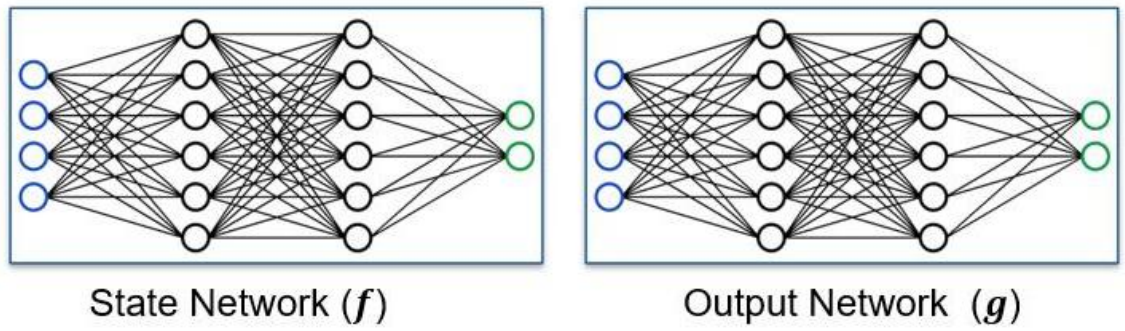
- Co je Neural State-Space
  - standardní state-space model

$$\dot{x}_t = f(x_t, u_t)$$
$$y_t = g(x_t, u_t)$$

x ... stav soustavy
y ... měřený výstup
u ... vstup
f, g ... nelineární funkce

  - neural state-space model má funkce *f* a *g* reprezentovány neuronovými sítěmi

$$\dot{x}_t = f(x_t, u_t)$$
$$y_t = g(x_t, u_t)$$



Neural State-Space model

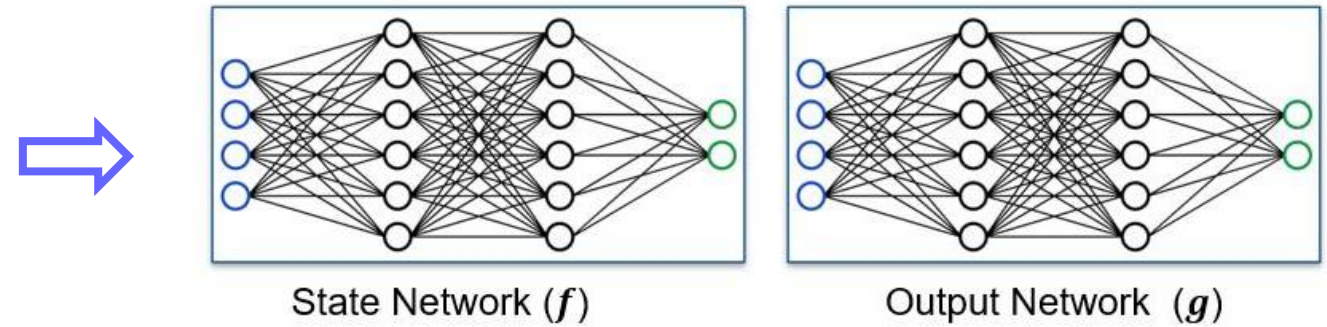
- Neural State-Space může být i diskrétní
  - standardní diskrétní state-space model

$$x_{k+1} = f(x_k, u_k)$$
$$y_k = g(x_k, u_k)$$

x ... stav soustavy
y ... měřený výstup
u ... vstup
f, g ... nelineární funkce

  - diskrétní neural state-space model má funkce *f* a *g* reprezentovány neuronovými sítěmi

$$x_{k+1} = f(x_k, u_k)$$
$$y_k = g(x_k, u_k)$$



Neural State-Space model

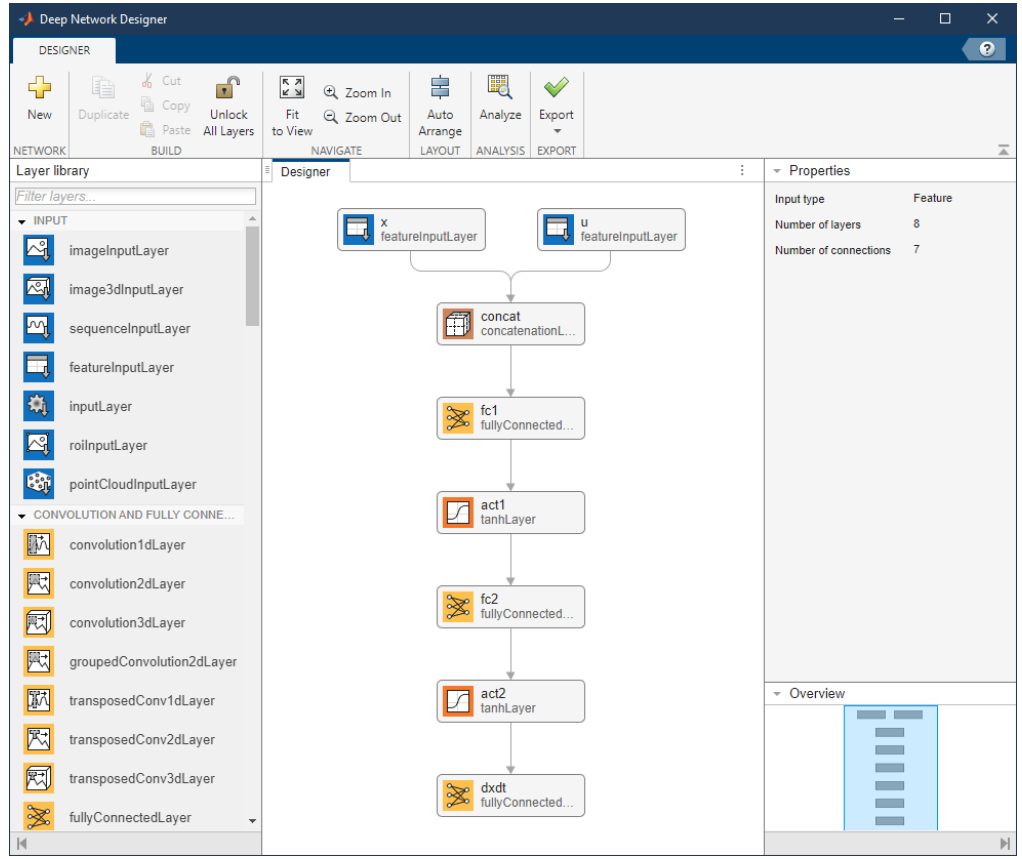
- Typická síť pro spojitý NSS
  - pro jeden stav, jeden vstup, výstup = stav

```
nss = idNeuralStateSpace(1,NumInputs=1)

nss =

Continuous-time Neural ODE in 1 variables
     dx/dt = f(x(t),u(t))
      y(t) = x(t) + e(t)

f(.) network:
  Deep network with 2 fully connected, hidden layers
  Activation function: Tanh
```



- Automatizované vytvoření sítí s uživatelskými parametry: `createMLPNetwork()`

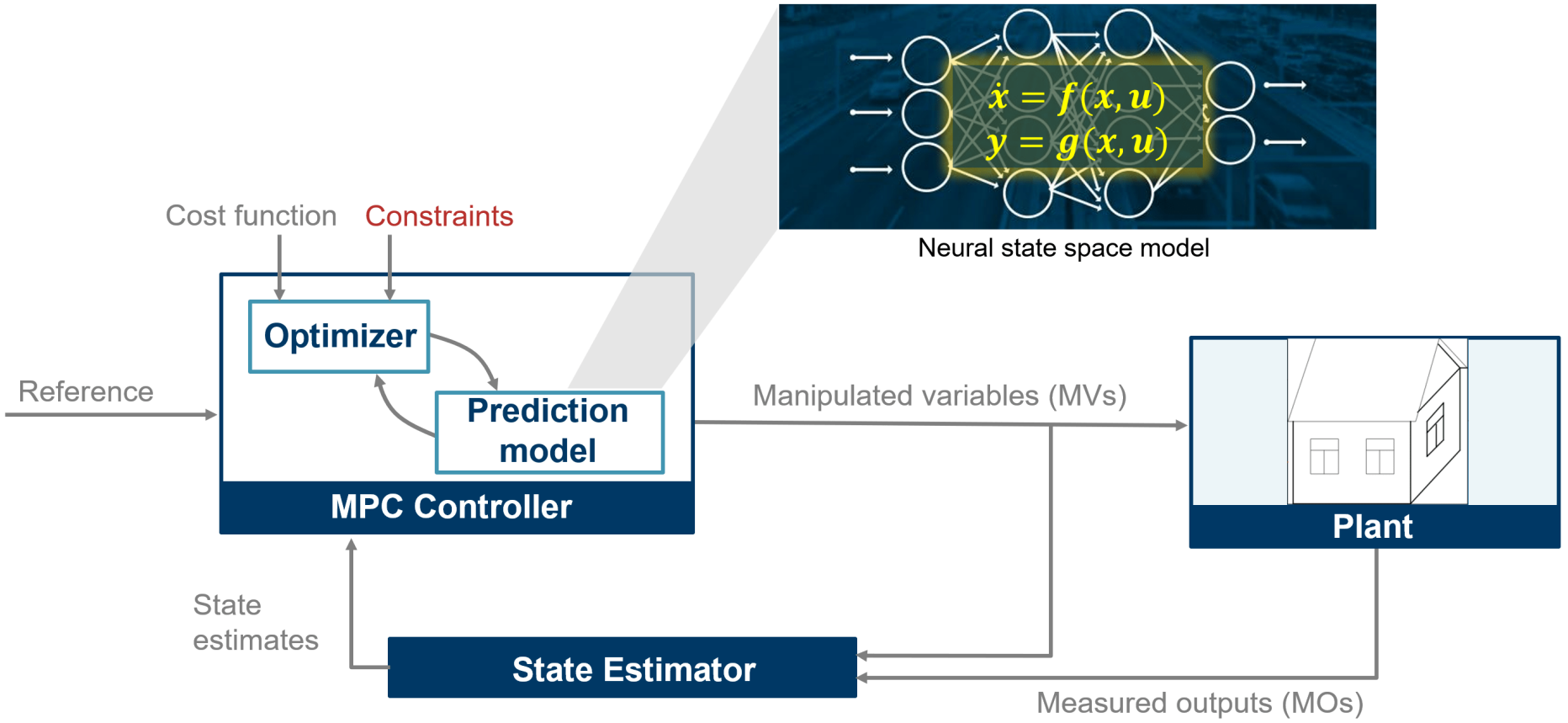
Identifikace Neural State-Space modelu a propojení s MPC

- Postup identifikace NSS modelu
  - získat data z reálné soustavy vstup-výstup: `iddata()`
  - vytvořit neural state-space objektu: `idNeuralStateSpace()`
  - vytvořit neuronové sítě pro aproximaci funkcí f a g: `createMLPNetwork()`
  - nastavit předvolby pro učení sítí: `nssTrainingOptions()`
  - identifikovat model (= učení sítí): `nlssest()`
- Propojení s nelineárním MPC:
  - generovat z NSS funkce pro odhad stavu a výpočet Jacobiánu: `generateMATLABFunction()`
  - použít funkce v návrhu MPC

Návrh nelineárního MPC

- Vytvoření objektu nelineárního MPC
  - obecný nelineární MPC: `nlmpc()`
  - vícestupňový nelineární MPC: `nlmpcMultistage()`
- Specifikace predikčního modelu
  - zadání funkcí f a g pro výpočet stavu a výstupu
  - pro lepší výkon specifikovat funkce pro výpočet jejich Jakobiánu
- Zadání optimalizačního kritéria, omezení, řešiče a dalších parametrů
- Volání MPC
  - volání objektu MPC z prostředí MATLAB: `nlmpcmove()`
  - simulace v prostředí Simulink: bloky pro volání MPC různých typů
- Generování kódu z MPC algoritmu a nasazení na cílovou platformu

Ukázka: Řídicí systém vytápění domácnosti s využitím MPC



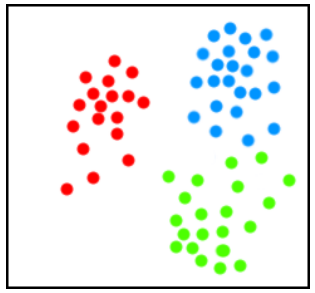
https://www.mathworks.com/help/mpc/ug/use-multistage-mpc-with-neural-state-space-prediction-model-for-house-heating.html

Reinforcement learning (RL)

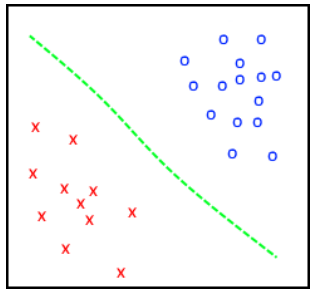
- Jedna z metod strojového učení (ML)

machine learning

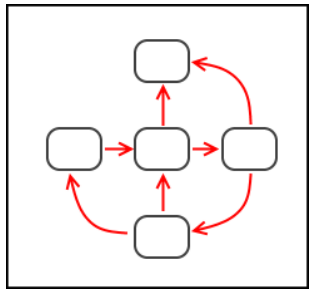
unsupervised learning
(neoznačená data)



supervised learning
(označená data)



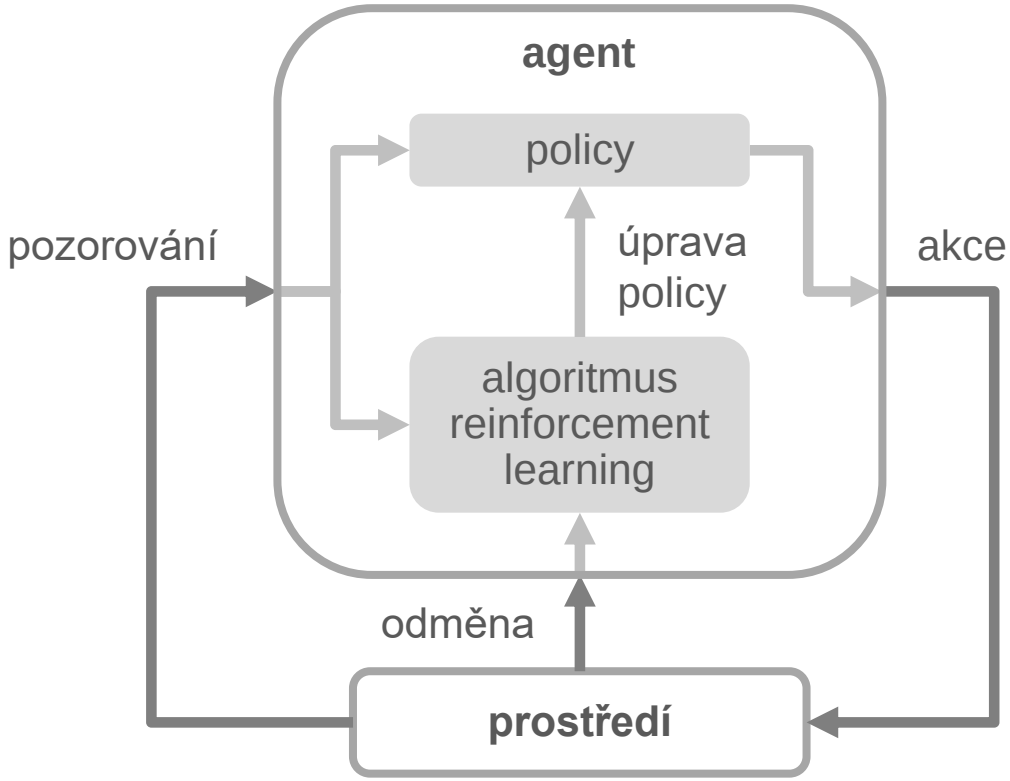
reinforcement learning
(data z interakce)



- Deep reinforcement learning
  - metoda reinforcement learning aplikovaná pro hluboké neuronové sítě

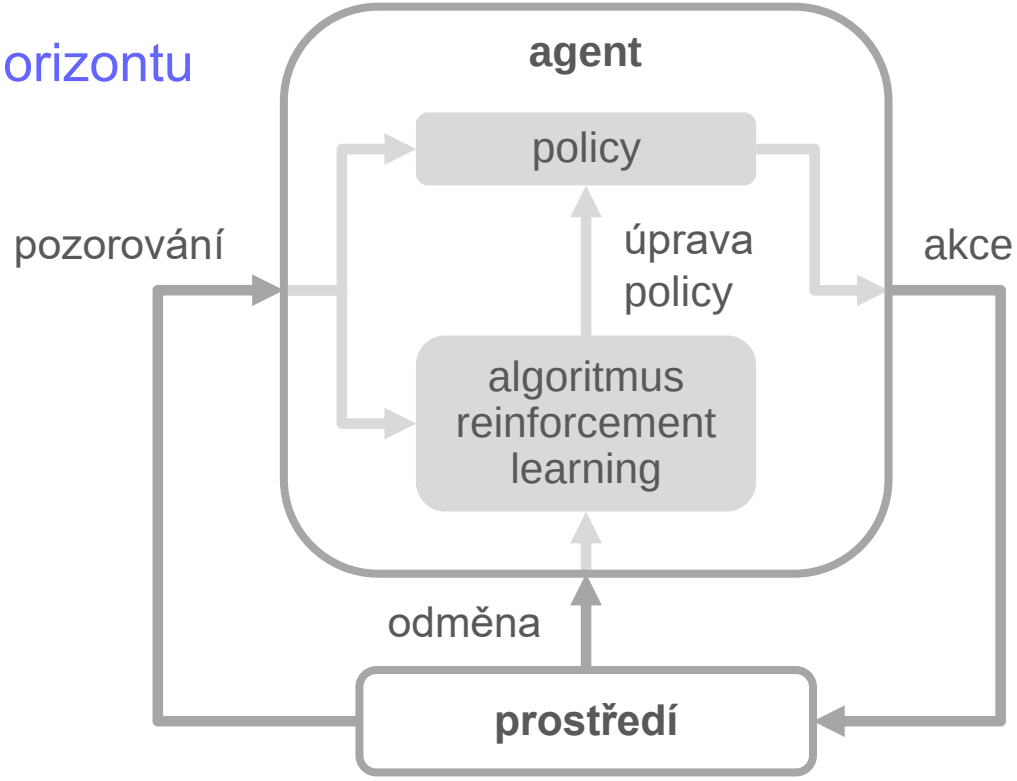
Reinforcement learning – hlavní části

- Agent
  - algoritmus policy (politika)
  - algoritmus učení metodou RL
- Prostředí
- Data
  - pozorování
  - akce
  - odměna



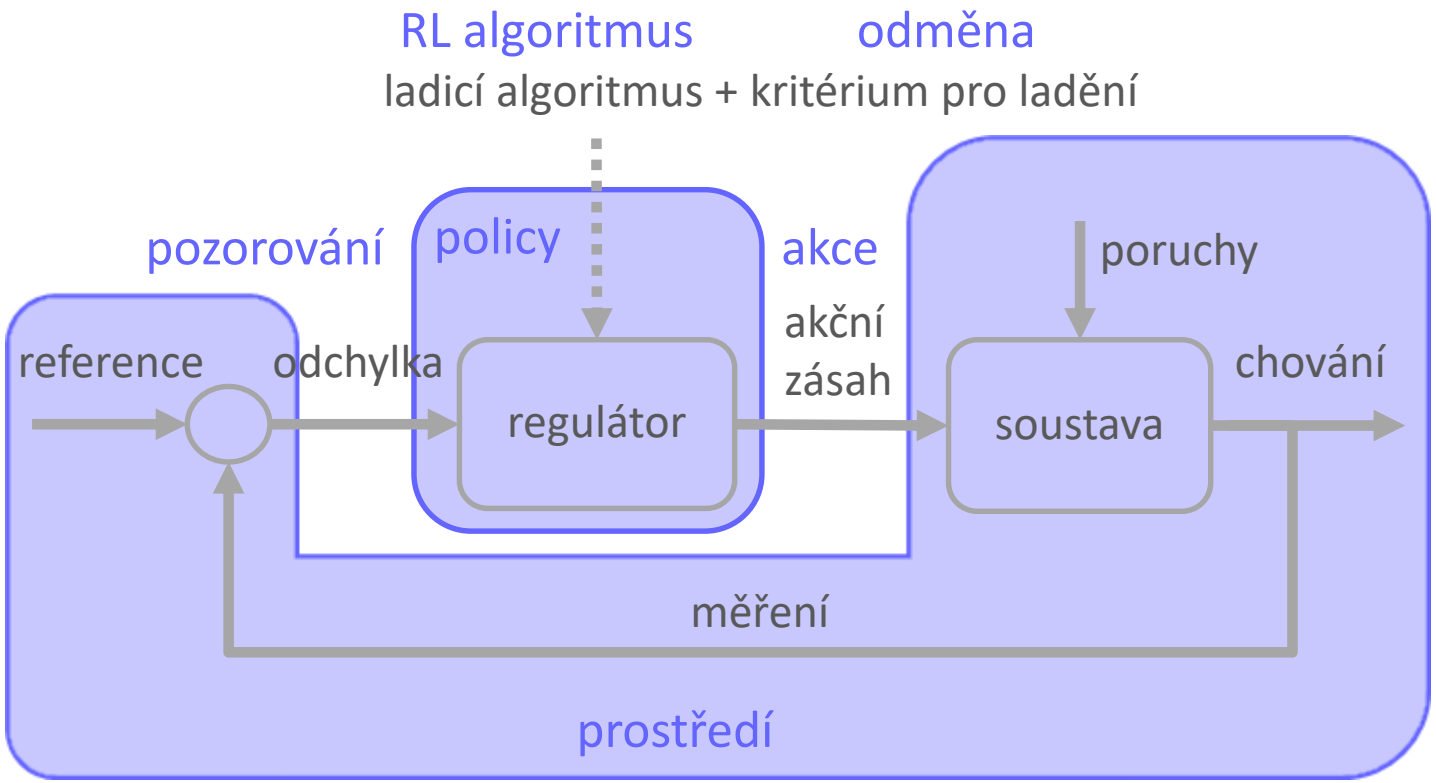
Reinforcement learning – jak funguje

- Princip
  - počítačový agent se učí optimálnímu chování opakovanou interakcí s dynamickým prostředím
- Cíl
  - maximalizovat odměnu v dlouhodobém časovém horizontu
- Algoritmus policy
  - hluboká neuronová síť (nejčastěji)
  - funkce: řídicí systém, rozhodovací algoritmus
- Použití
  - kde jsou tradiční metody obtížně formulovatelné
  - pro obtížně interpretovatelné signály (např. obraz)
  - autonomní systémy, robotika, ...

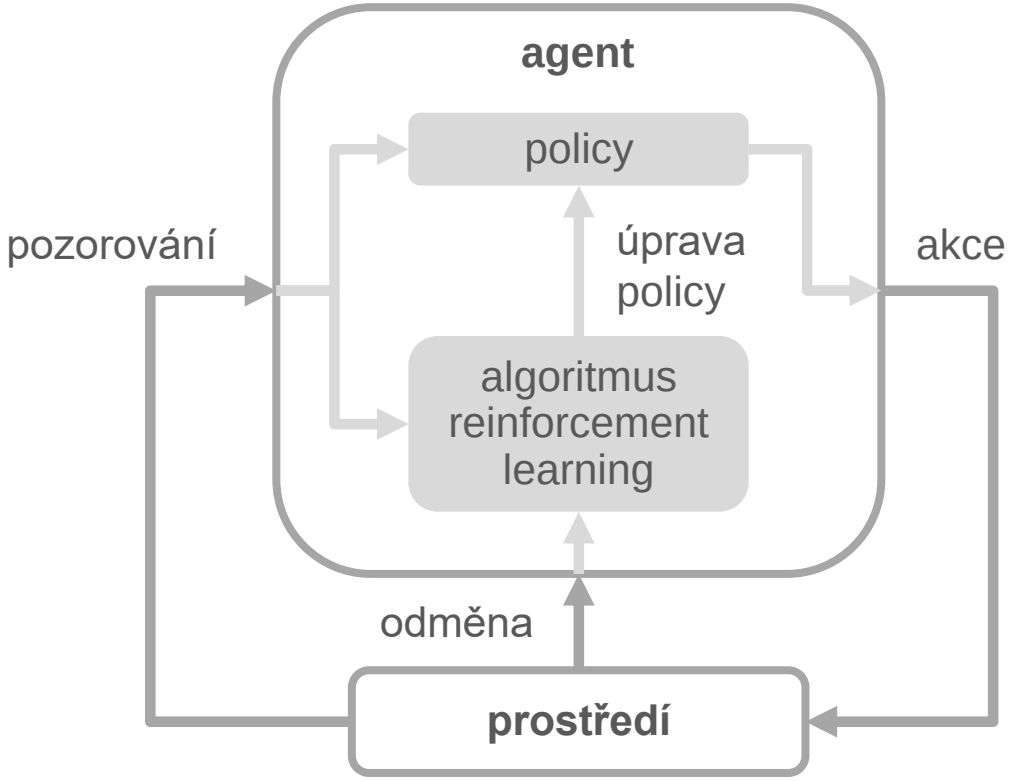


Porovnání RL s tradičními řídicími systémy

Tradiční řídicí systém

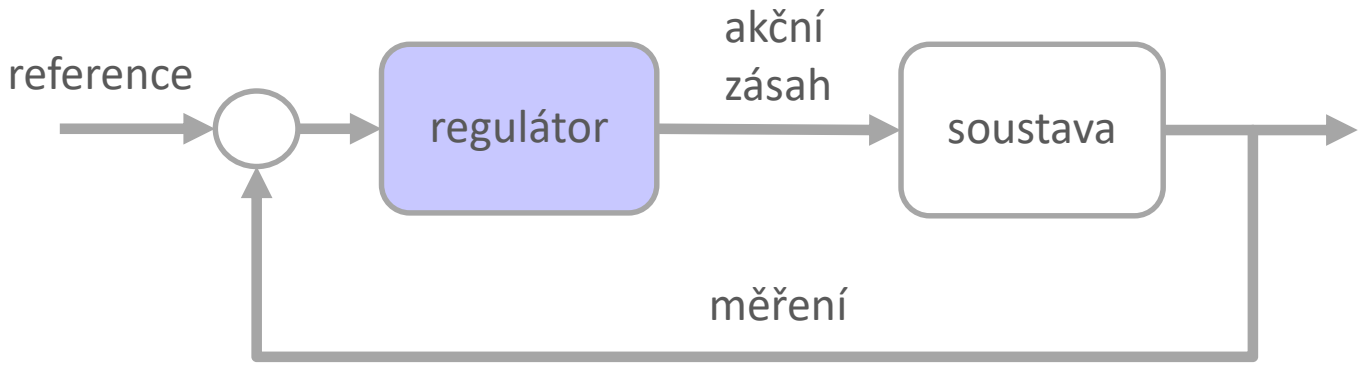


Systém reinforcement learning



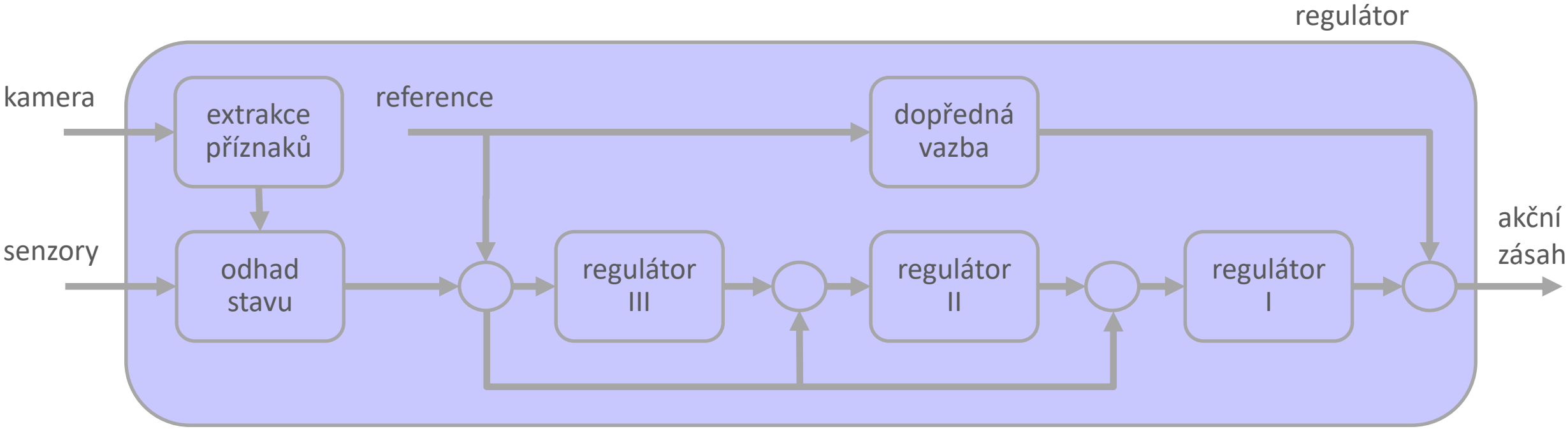
Tradiční řídicí systémy

- Pro jednoduché systémy funguje dobře



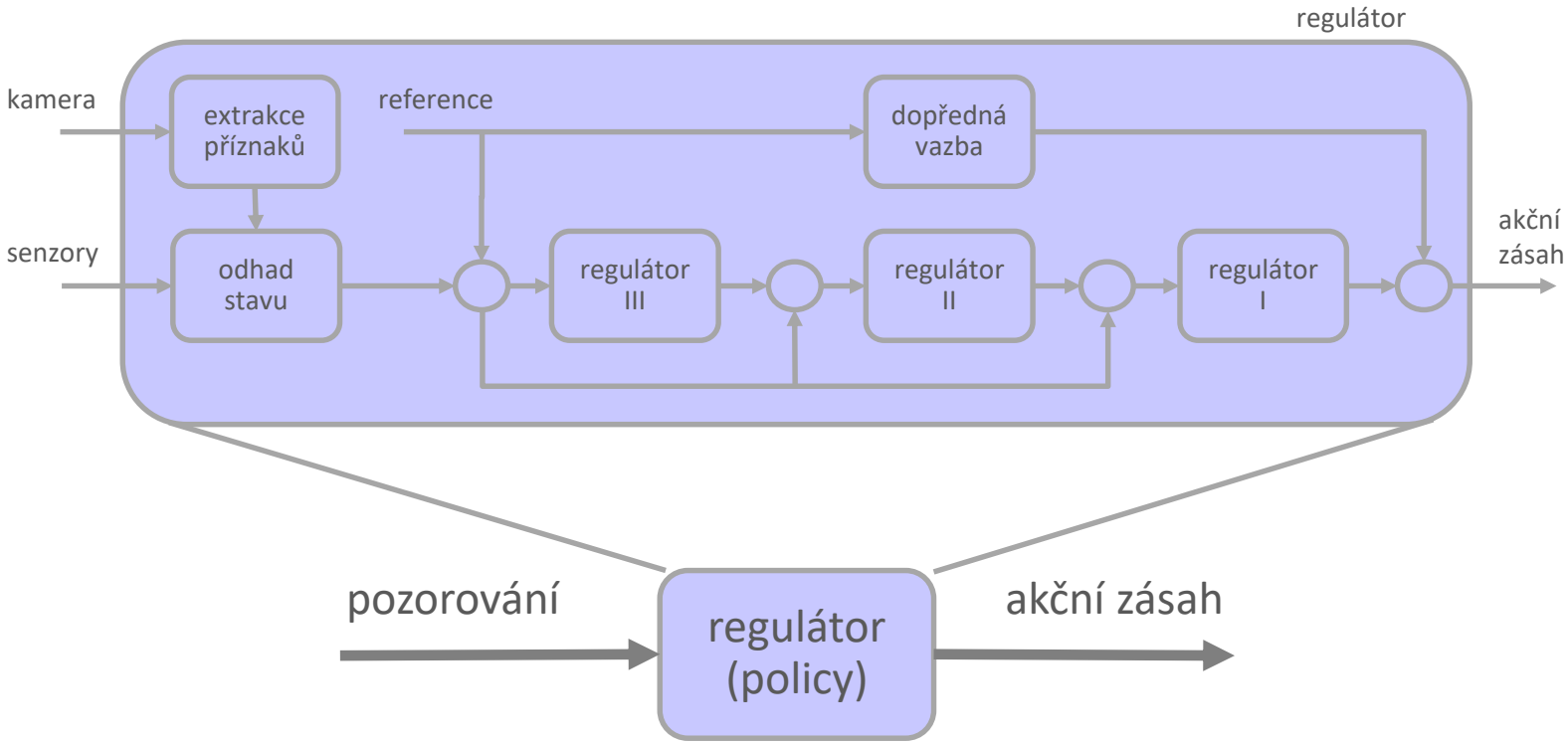
Tradiční řídicí systémy

- Pro komplexní systémy
  - rozdělení do menších částí řešených nezávisle
  - několik řídicích smyček, kaskádové řízení, kompenzační vazby, atd.
  - úskalí: volba vhodné architektury, provázání smyček při ladění



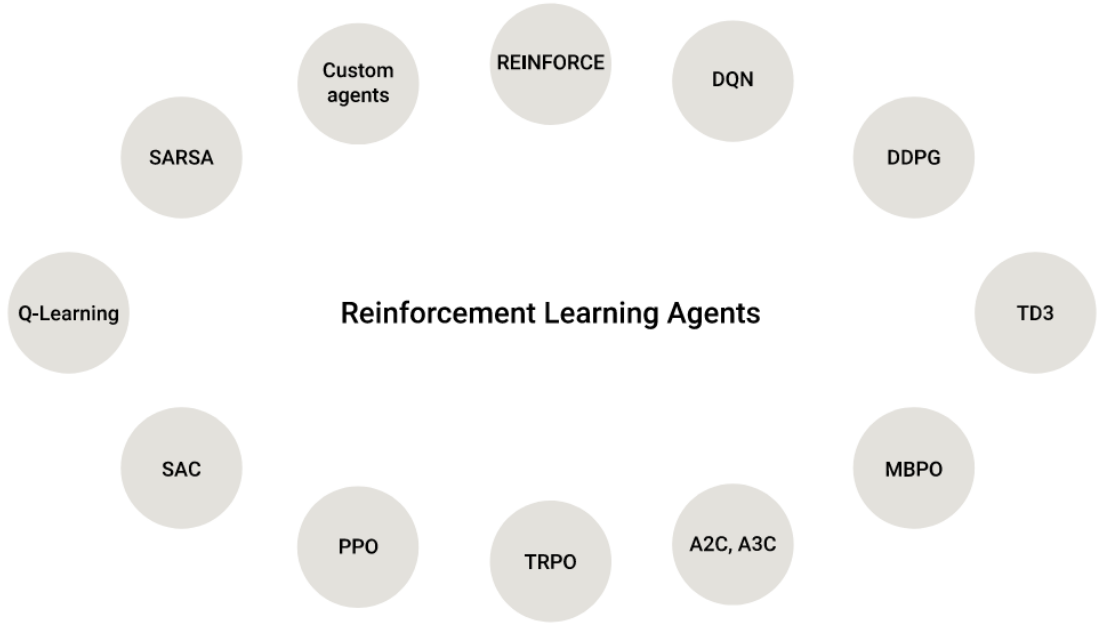
Alternativa: reinforcement learning

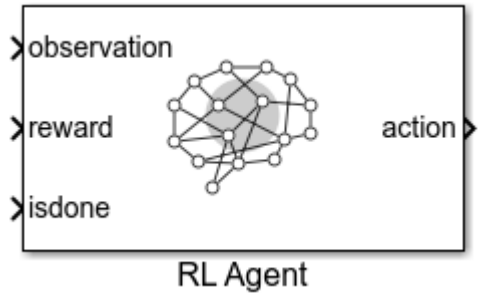
- Sloučení celého řídicího systému do jediné funkce
  - vstupem jsou všechna pozorování
  - výstupem je přímo akční zásah na nejnižší úrovni



Reinforcement learning agent

- Algoritmus „policy“
  - mapování vstupního stavu na výstupní akci
- Vnitřní reprezentace
  - hluboká neuronová síť
- Algoritmus „reinforcement learning“
  - úprava policy
  - maximalizuje odměnu v dlouhodobém časovém horizontu
  - k dispozici mnoho různých algoritmů
- Implementace
  - objekt v prostředí MATLAB, blok v prostředí Simulink



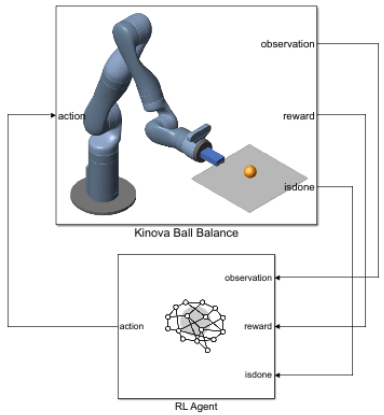


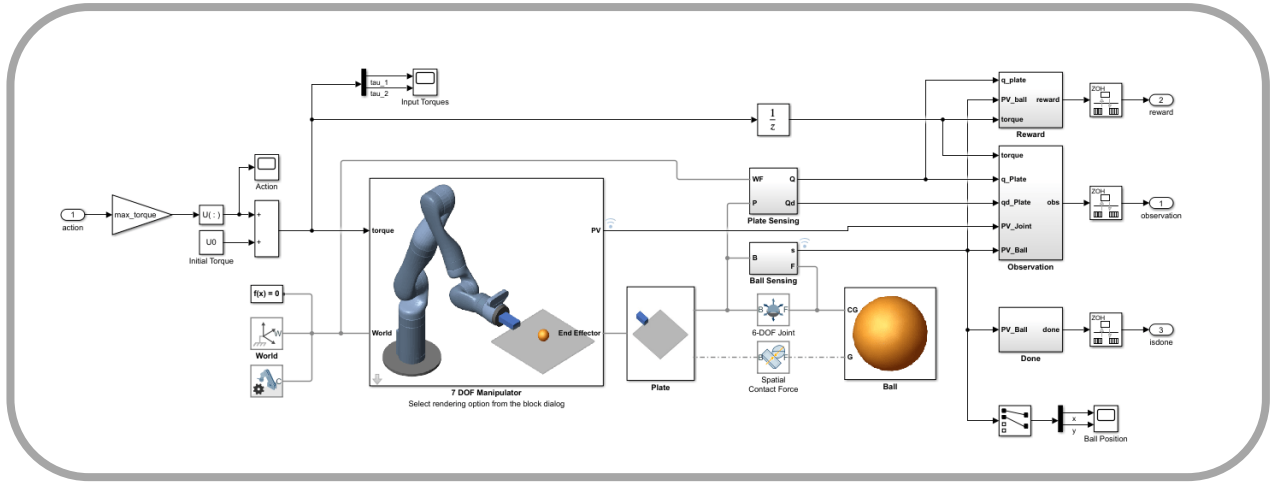
Prostředí

- Systém (soustava) a jeho okolí využitý k učení
- Učení
  - dynamický proces
  - agent interaguje s prostředím
- Výpočet odměny
  - ohodnocení přínosu akce
- Učení s reálným systémem
  - nákladné, pomalé, rizikové
- ⇨ využití simulovaného prostředí

Model prostředí

- Simulační model *prostředí* vytvořený v nástroji Simulink / Simscape
  - dynamický model systému a jeho okolí
  - lze využít nástroje pro fyzikální modelování
  - pozorování, akce a výpočet odměny součást modelu

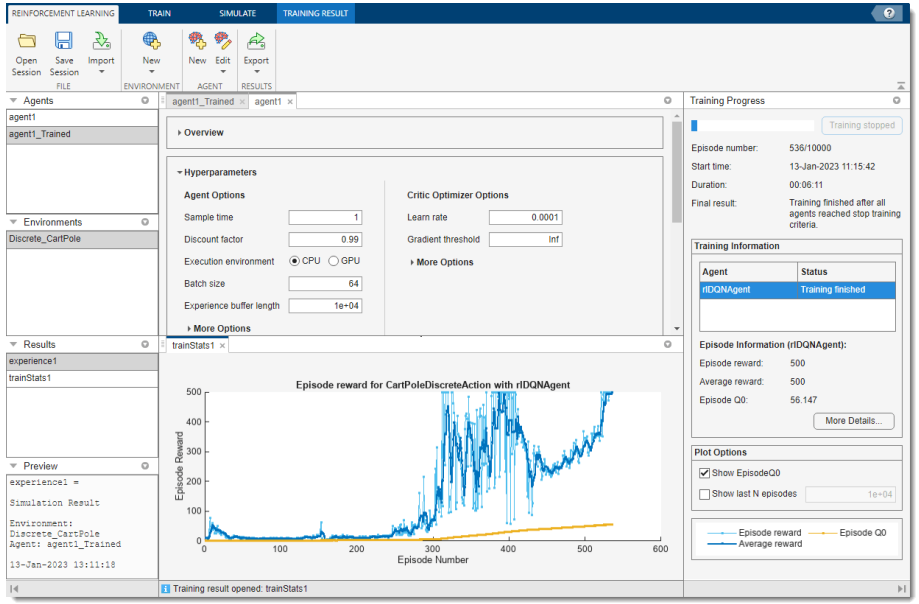


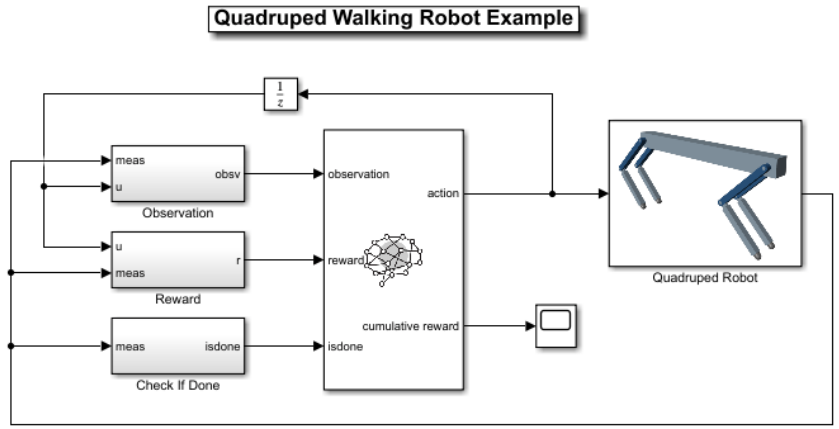


- Model *prostředí* vytvořený prostředky jazyka MATLAB
  - funkce a třídy v jazyce MATLAB

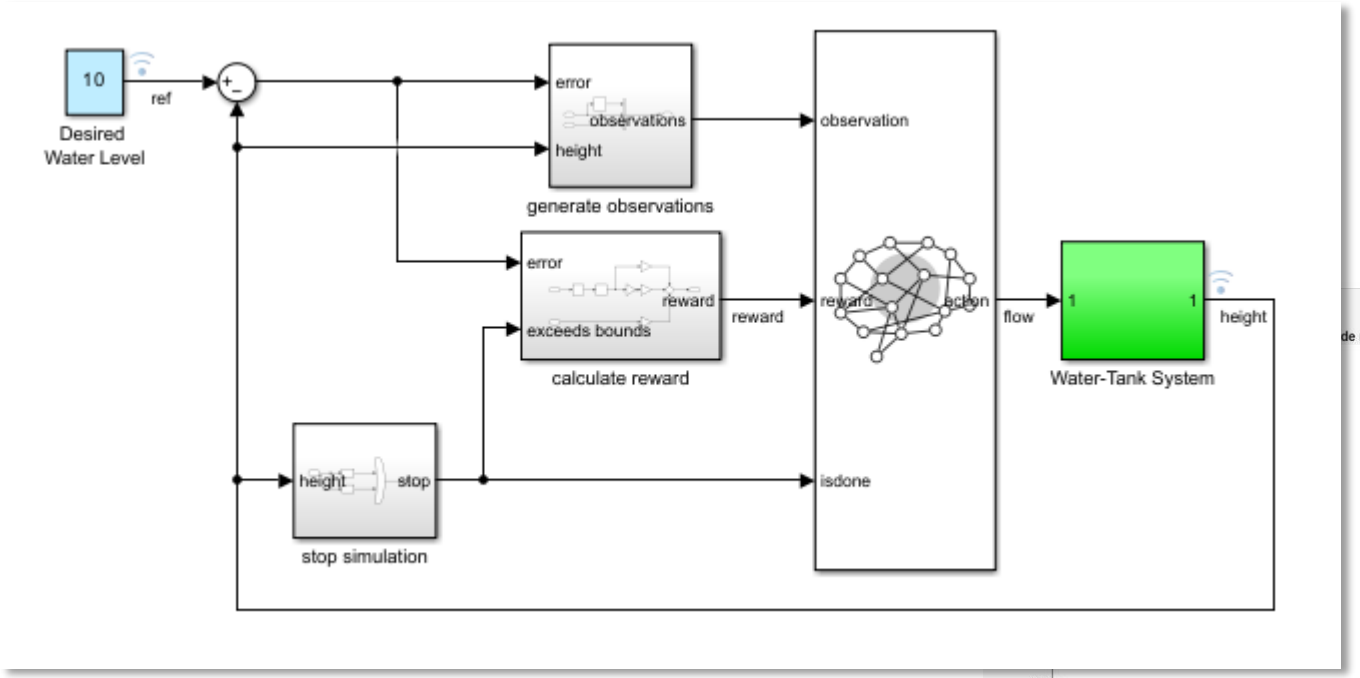
Nástroje pro (deep) reinforcement learning

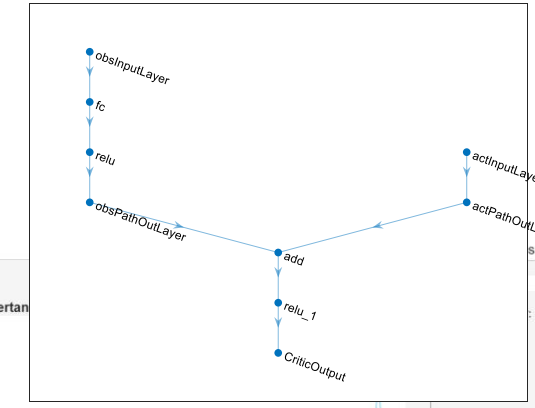
- Reinforcement Learning Designer app
  - interaktivní návrh a učení agenta
  - průvodce výběrem typu agenta
  - výběr RL algoritmu
- Připravené funkce a objekty
  - návrh a učení pomocí příkazů
- Bloky v prostředí Simulink
  - RL Agent – simulace a učení agenta
  - Policy – simulace a nasazení naučené funkce policy

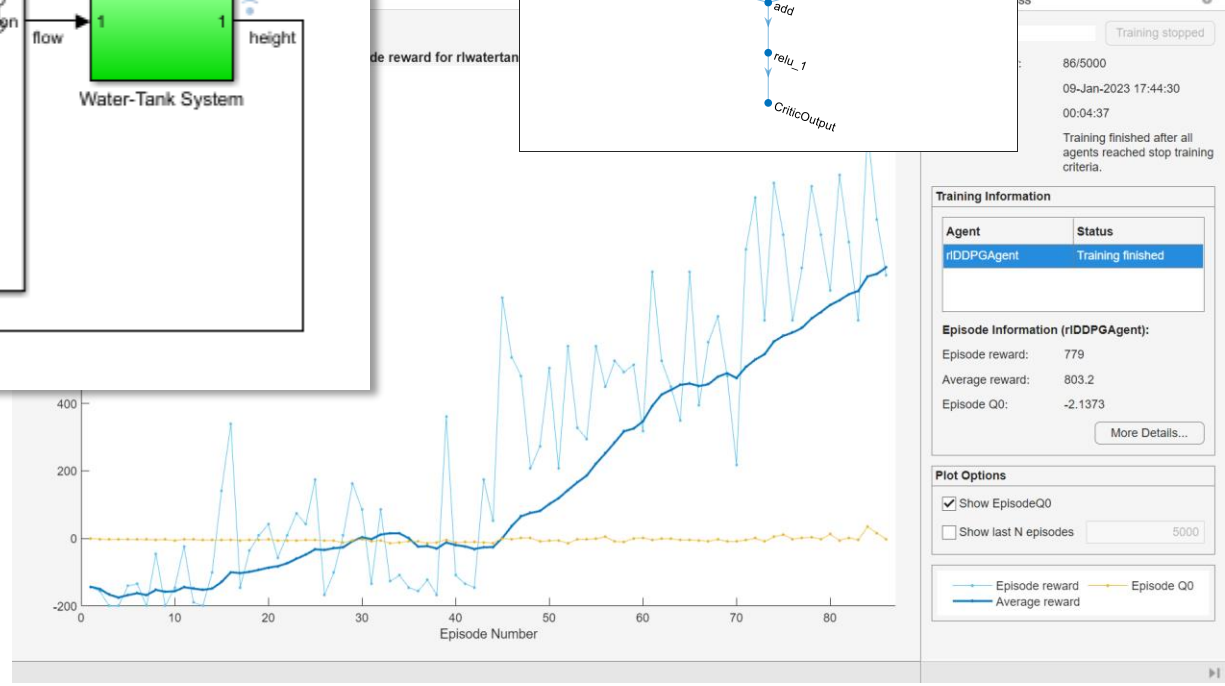




Ukázka: Řízení hladiny v nádrži metodou RL



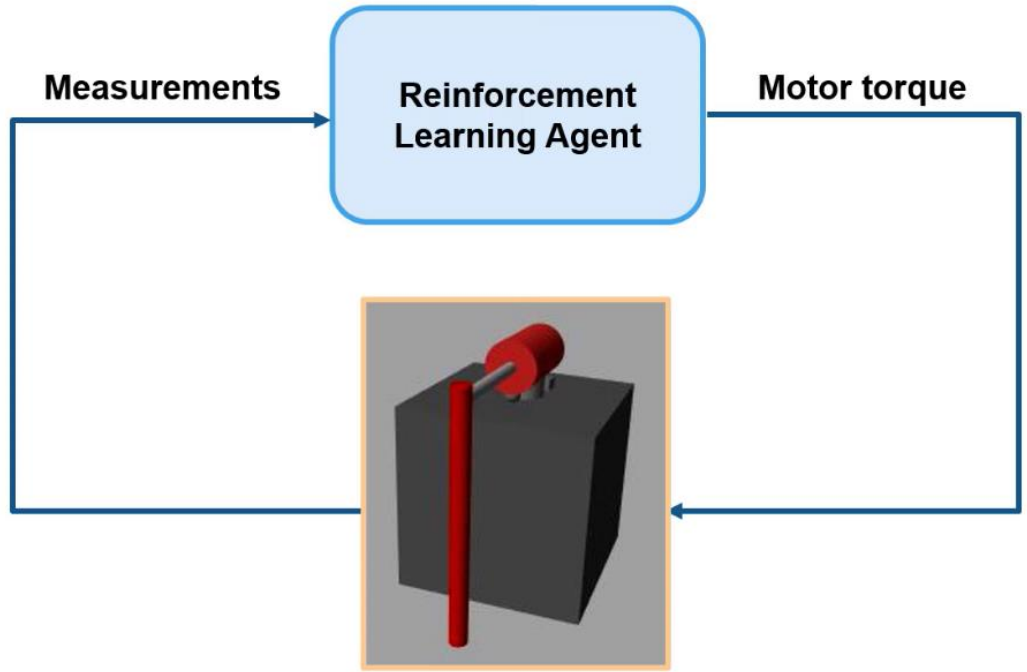




https://www.mathworks.com/help/reinforcement-learning/ug/create-simulink-environment-and-train-agent.html

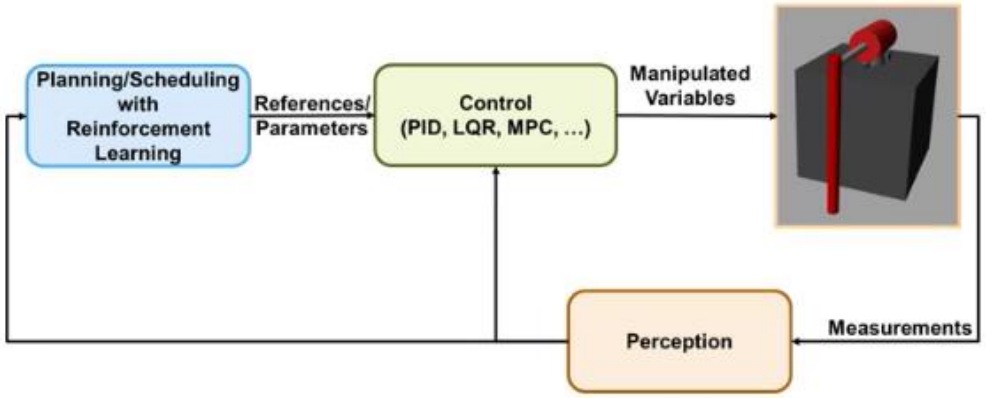
Modifikace použití RL pro složité systémy

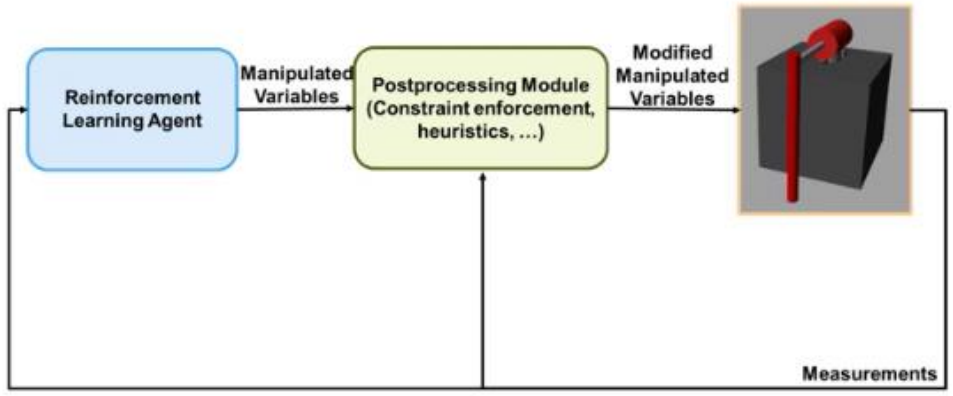
- Využití RL agenta pro kompletní („end-to-end“) řízení může mít některá úskalí
  - RL agent je „black-box“
  - obtížné dosáhnout správného naučení
    - s ohledem na všechny aspekty řízení
    - vnímání + plánování + vlastní řízení
  - nelze formálně zaručit správné chování

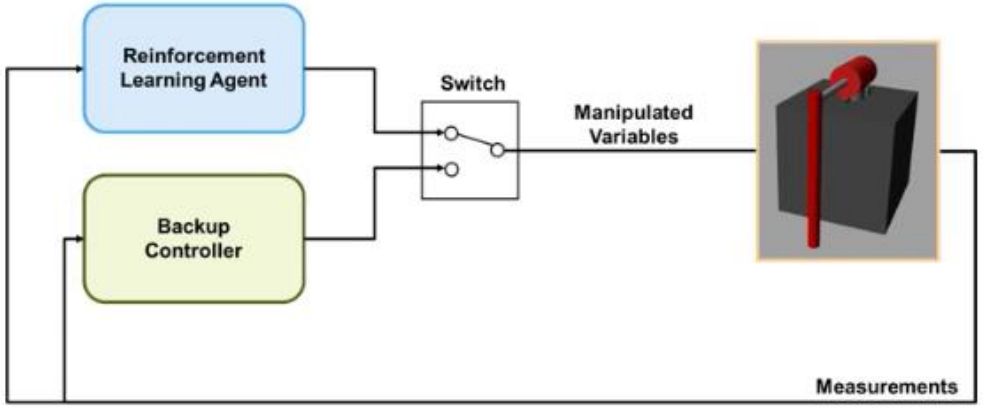


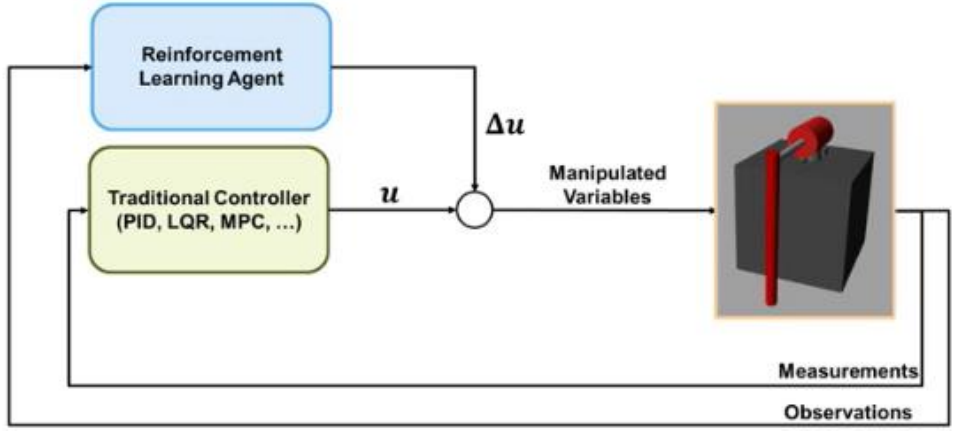
Modifikace použití RL pro složité systémy

- Řešení – kombinace RL a tradičních metod:



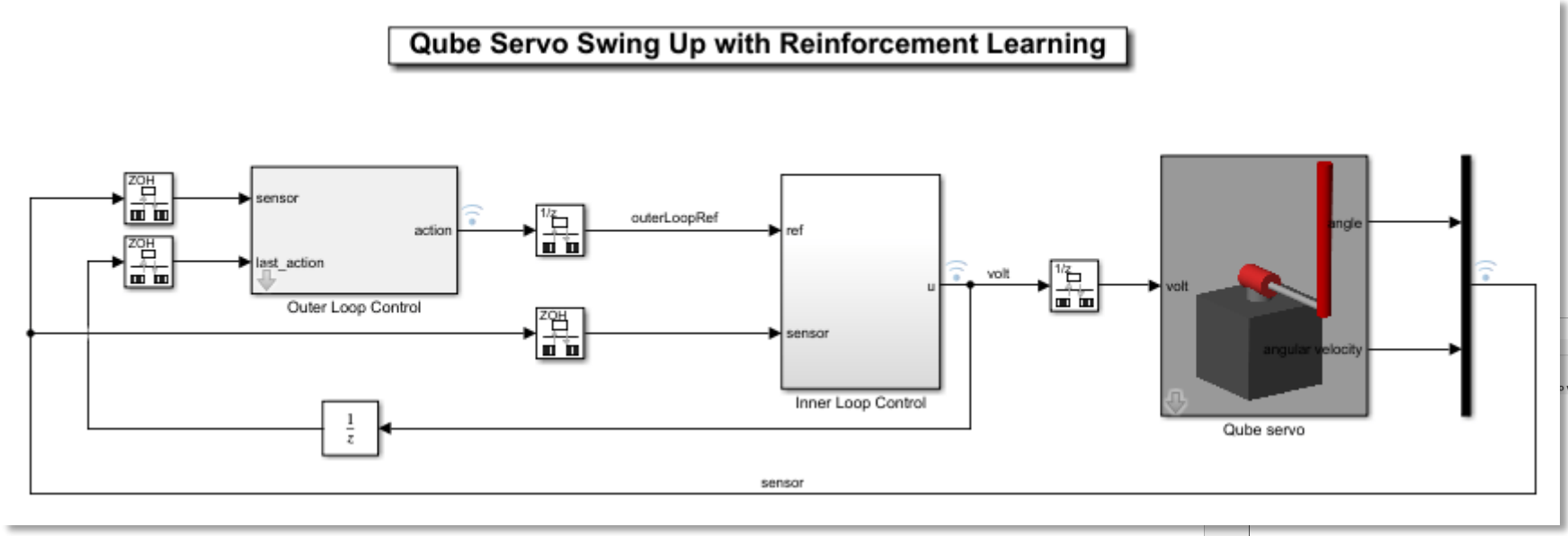


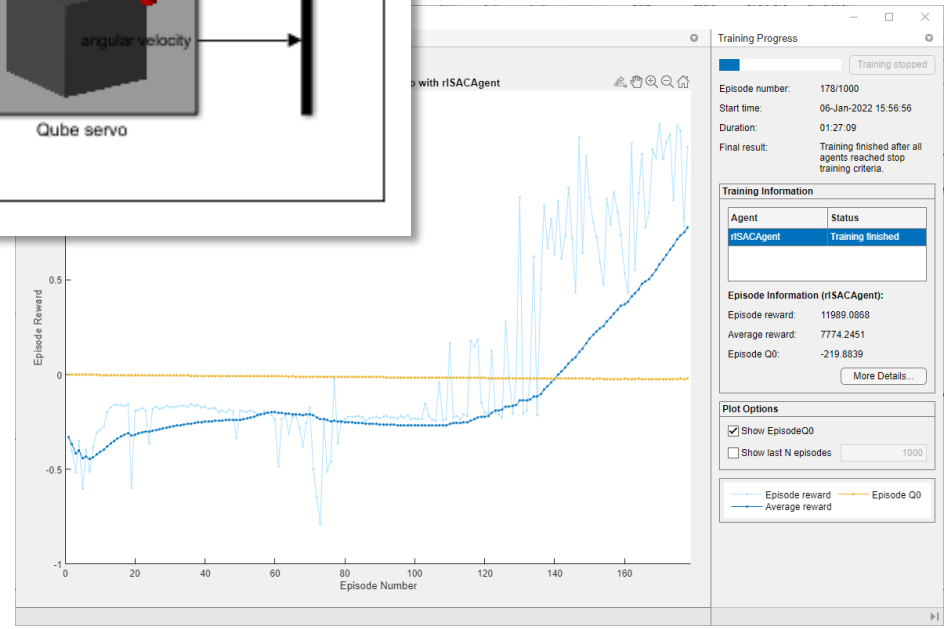


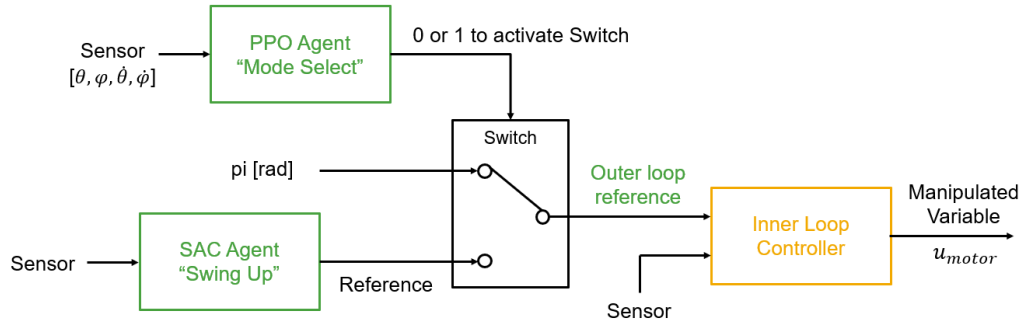


Practical Reinforcement Learning for Controls: Design, Test, and Deployment

Ukázka: Řízení inverzního kyvadla







https://www.mathworks.com/help/reinforcement-learning/ug/train-agents-to-control-quanser-qube-pendulum.html

Otázky