Zoznam skratiek a značiek

CaaS - Controller as a Service - regulátor ako služba
IoT - Internet of Things - internet vecí
MIMO - multiple input multiple output - viacrozmerný systéme
MPC - Model predictive controller - prediktivný regulátor
MVC - Model-View-Controller štruktúra aplikácie
PID - regulátor s Proporčnou, Integračnou a Derivačnou zložkou
RAM - Random access memory - operačná pamäť
REST - Representational state transfer architectural style
SISO - single input single output - jednorozmerný systém

Úvod

V súčasnosti existuje mnoho metód automatického riadenia systémov. Od klasických metód ako je PID regulátor, cez rôzne pokročilé metódy ako je robustné, adaptívne alebo aj prediktívne riadenie. Predmetom tejto práce je

- spomenúť hlavné výhody prediktívnej metódy a jej súčasný stav.
- Predmetom práce je tiež definovať pojem Internet vecí (Internet of Things - IoT) ako nový typ architektúry informačných systémov a jeho špecifiká.
- V praktickej časti je spojenie predošlých dvoch bodov v myšlienke regulátor ako služba (Controller as a Service - CaaS) a jej implementácia.

Motivácia k MPC

Prediktívne riadenie (MPC, model predictive controller) je pokročilá metóda riadenia založená na optimalizácii, ktorá bola využívaná na aplikáciu v systémoch s pomalou dynamikou, napríklad v chemických, či petrochemických procesoch. Na rozdiel od lineárno-kvadratického regulátora, MPC na optimálne riadenie ponúka explicitné ošetrenie procesných obmedzení, ktoré vznikajú z prirodzených požiadaviek, napríklad efektivita nákladov, bezpečnostné obmedzenia akčných členov a iné.[14]

Hlavná výhoda regulátora je, že je riešený ako optimalizačný problém, takže sa snaží minimalizovať okrem iného hlavne potrebný riadiaci zásah na dosiahnutie žiadaného výstupu riadeného systému. Ďalšia výhoda pri riešení optimalizačného problému je, že sa jednoducho môžu zadať ohraničenia systému. Rôzne obmedzenia môžu byť aplikované na riadený systém a napriek tomu môže riaditeľný s minimálnym riadiacim zásahom. Táto metóda riadenia môže byť prirovnaná k prepočítavaniu ťahov v šachu. Pri prepočítavaní sa ráta s tým, čo sa môže udiať v čase v závislosti od poznania procesu, pri šachovej hre podľa jej pravidiel a podľa toho optimalizovať riadiace zásahy, ťahy v šachu, aby sa dosiahol najlepší výsledok z dlhodobého hľadiska, v prípade šachu, to je vyhratá partia. Pri MPC regulátoroch sa dá predísť javom, ktoré sa vyskytujú pri konvenčných regulátoroch ako PID, ako sú riadiace zásahy, ktoré dosahujú dobré výsledky z krátkodobého hľadiska, ale v konečnom dôsledku sa prejavia ako vysoko nákladné. Tento jav môže byť pomenovaný ako „vyhrať bitku, ale prehrať vojnu“.

Ďalšie výhody MPC regulátora sú také, že jednoducho vedia riadiť viac premenné systémy a ako už bolo spomenuté, zahrnúť obmedzenia systému – výstupu, riadiaceho zásahu, stavu systému, už pri návrhu regulátora. MPC regulátor obsahuje viacero pre-

menných, pomocou, ktorých je možné ho vyladiť takmer pre každý proces.

Medzi nevýhody MPC regulátora patrí napríklad to, že niektoré MPC modely sú limitované len na stabilný proces v otvorenej slučke. Často vyžadujú veľký počet koeficientov modelu na opis odozvy systému. Niektoré MPC modely zase sú formulované na rušenie na výstupe a tie by ťažko mohli zvládnuť poruchy na vstupe. Niektoré MPC modely sú zase upravované na výstupe, pretože model nie je totožný s reálnym systémom. Tieto modely sú zvyčajne upravené o konštantu, podľa nameraných údajov, nerátajú však s tým, že táto zmena na výstupe sa môže v budúcnosti zmeniť, čo môže mať za následok, že finálny výsledok nebude optimálny. Taktiež ak horizont predikcie nie je zvolený správne, tak riadenie nebude optimálne aj keď model systému správny bude. Niektoré systémy majú širokú škálu prevádzkových podmienok, ktoré sa často menia. Medzi príklady patria exotermické reaktory, procesy na dávkové spracovanie a tiež systémy, kde rôzny spotrebitelia majú rôzne špecifikácie produktov. Lineárne modely MPC regulátorov nie sú schopné zvládnuť dynamické správanie týchto procesov, preto musí byť použitý nelineárny model pre lepšie riadenie.[10]

Motivácia k IoT

Aktuálne je pojem IoT čoraz častejšie skloňovaný na konferenciách akademickej a rovnako aj komerčnej sféry. Rôzne popredné spoločnosti ako IDC, Gartner ai. zaoberajúce sa výskumnými a poradnými činnosťami v oblasti informačných a komunikačných technológií robia odhady využitia. Tvrdenie portálu www.crn.com: „Napriek tomu, že IoT bol vždy trochu vágny pojem pri špecifikácii obchodných partnerov a produktov, ktoré už sú na trhu, analytici predikovali, že pripojené zariadenia (connected devices) majú veľký potenciál príležitosti, ktoré budú výnosné. Júnový prieskum trhu spoločnosti IDC predikoval, že výdavky na IoT dosiahnú v roku 2020 sumu vo výške 1,7 biliónov dolárov, zatiaľ čo Gartner predpovedal, že v tom istom roku bude pripojených 21 miliárd zariadení.“[1] Na základe tohto a ďalších podobných článkov je teda motivácia hľadanie vhodných prípadov využitia IoT.

Okrem toho treba zopakovať, že IoT je spojenie minimálne dvoch rozsiahlych technických odborov, neberúc do úvahy spoločenské, prírodne ani lekárske vedné odbory, ktoré tiež môžu skúmať dopad IoT na ne. Rozsiahlosť tejto problematiky je určite nepopierateľná. Druhá motivácia teda je získavanie nadhľadu nad spleťou vznikajúcich a zanikajúcich technológií a štandardov.

Ostatná motivácia k skúmaniu IoT je porovnanie rozdielov pri návrhu, vývoji a správe oproti klasickým čisto softvérovým informačným systémom.

Motivácia k CaaS

Napriek tomu, že aktuálny trend vo vývoji hardvéru je znižovanie rozmerov a zvyšovanie výkonu, online prediktívny regulátor je stále náročný na výpočtový výkon. Preto popri výskumoch aplikovania offline metódy prediktívneho algoritmu na zariadenia blízke hardvérovej úrovni napríklad FPGA hradlá, vznikla myšlienka implementácie online MPC regulátora na server - „do cloudu“, ako službu, kde je možné zabezpečiť takmer neobmedzený výkon a jedinou prekážkou môže byť rýchlosť sieťového pripojenia. Realizácia tejto myšlienky je implementovaná v prostredí inteligentnej budovy.

1 MPC regulátor

Ako už bolo v úvode spomenuté, práca spája viacero odborov do jednej implementácie. Táto časť preto je venovaná návrhu a overenia MPC regulátora.

1.1 Teoretický základ MPC

V tejto časti sa vysvetlí história, matematický základ a variácie MPC regulátora.

1.1.1 On-line MPC

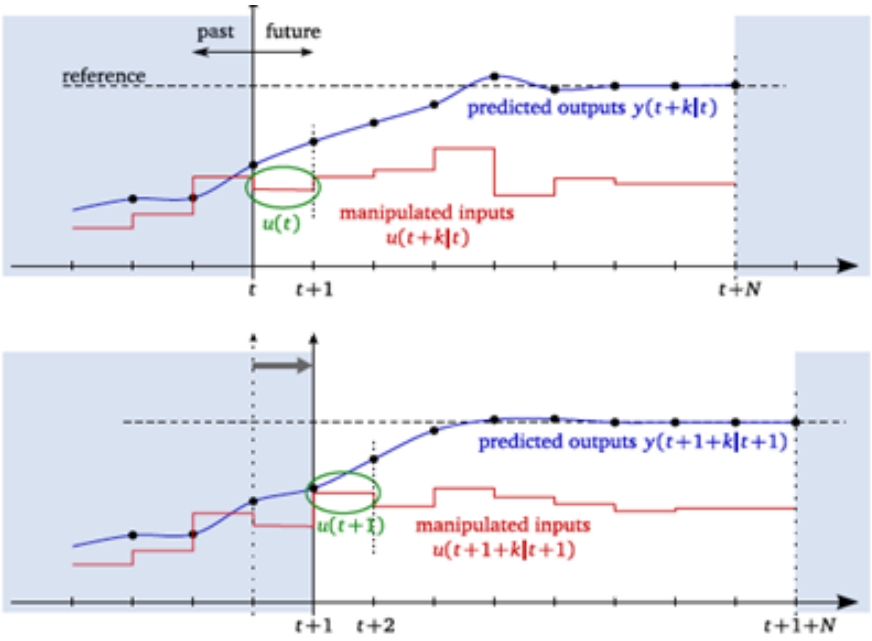
Prediktívny regulátor je, na najnižšej úrovni, metóda riadenia dynamických systémov, ktorá využíva nástroje matematickej optimalizácie. Spoločné črty všetkých prístupov riešenia problému prediktívnych regulátorov je vypočítať on-line, v každej časovej vzorke, optimálny riadiaci zásah v konečnom horizonte predikcie pre dynamický model systému, kde aktuálny stav je počiatočný stav. Iba prvý element z vypočítanej sekvencie predikovaných riadiacich zásahov je potom aplikovaný na systém. V ďalšom okamihu vzorkovania je horizont predikcie posunutý a výpočet optimálneho riadiaceho zásahu je vykonávaný znovu pre novonadobudnutý stav. Táto myšlienka nie je nová, už v článku od Lee a Markus (1967), je možné nájsť nasledujúce tvrdenie: „Jedna technika na návrh regulátora so spätnou väzbou zo znalosti regulátora pre otvorenú slučku je merať aktuálny stav riadenia procesu a veľmi rýchlo vypočítať funkciu riadenia pre otvorenú slučku. Prvá časť tejto funkcie je potom využitá počas krátkeho intervalu, po ktorom je opäť zmeraný stav procesu a k nemu vypočítaná riadiaca funkcia pre otvorenú slučku. Táto procedúra je potom opakovaná.“ Technika popisovaná v článku od Lee a Markus (1967) je zvyčajne označovaná ako „Receding Horizon Control“ (RHC) – „Postupujúci horizont riadenia“ a dnes je viac-menej používaný ako synonymum k pojmu„Model Predictive Control“ – „Prediktívne riadenie“. [8] Popisovaný princíp je znázornený na obrázku 1. Nech existuje lineárny dynamický model s časovo invariantnými parametrami, vyjadrený diskrétnou stavovou rovnicou:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \\ x(t) \in R^n, y(t) &\in R^P, u(t) \in R^m, \\ A \in R^{(n\times n)}, B &\in R^{(n\times m)}, C \in R^{(p\times n)} \end{aligned} \tag{1}$$

x, y, u sú stavy systému, výstupy systému a riadiace zásahy v uvedenom poradí v čase alebo lepšie povedané vo vzorke t.

n – predstavuje počet stavov systému

p – predstavuje počet výstupov



Obrázok 1: Postupujúci horizont riadenia

m – predstavuje počet vstupov

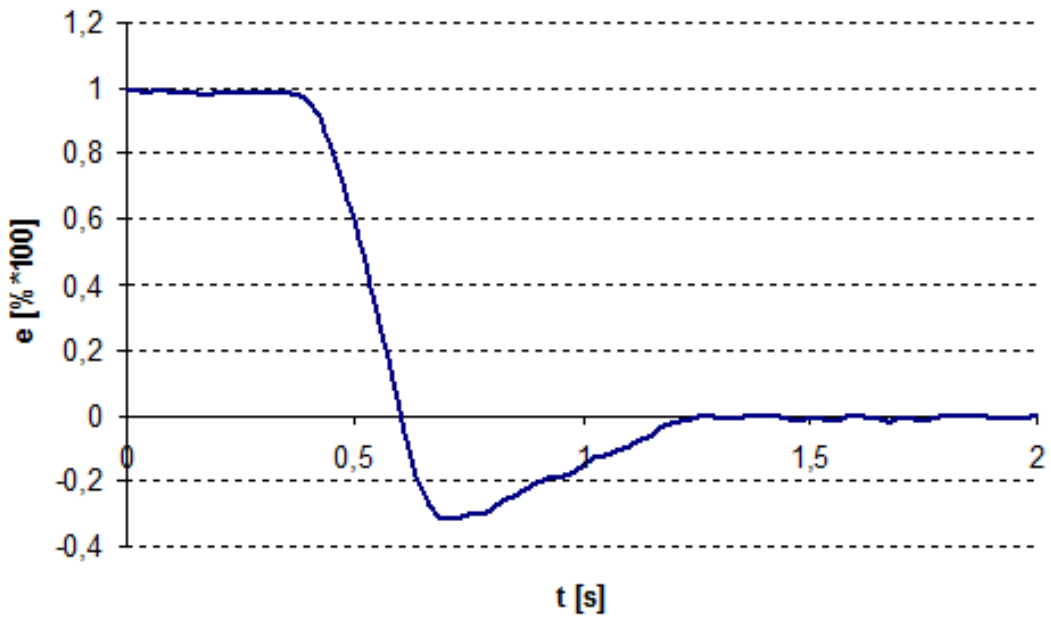
Matice A, B, C sú systémová matica, vstupná matica a výstupná matica v uvedenom poradí. System 1 musí spĺňať nasledujúce obmedzenia na stav a vstup:

$$\begin{aligned} x(t) \in X, u(t) \in U \\ U \subset R^m \\ X \subset R^n \end{aligned} \tag{2}$$

kde obmedzujúca množina riadiacich zásahov U je konvexná, kompaktná (uzavretá a ohraničená) a obmedzujúca množina stavov X je konvexná a uzavretá. Pre obidve množiny U a X sa predpokladá, že obsahujú počiatok v ich vnútri.[9] Najskôr je jednoduchšie vyriešiť optimálne riadenie bez obmedzení. Pozornosť bude upriamená na nájdenie takej optimálnej postupnosti u^*(k), ..., u^*(k+N – 1), ktorá bude minimalizovať zvolené kvadratické kritérium a zároveň rešpektovať dynamiku systému. Čo inými slovami znamená, že tento regulátor sa bude snažiť minimalizovať stav a rovnako riadiaci zásah, ešte lepšie povedané ich kvadrát. Ak by teda nebola špecifikovaná referenčná hodnota, ktorú má výstup regulátora sledovať, tak implicitne výstupná veličina prediktívneho regulátora konverguje k hodnote 0 alebo pri systémoch s viacerými výstupmi k nulovému vektoru.

N – predstavuje horizont predikcie. Ak je teda systém v stave x_0, ktorý poznáme a horizont predikcie je 3, tak do optimalizačnej funkcie vstupujú premenné x_1, x_2, x_3 a u_1, u_2, u_3 kde u_1 zabezpečí prechod do stavu x_1, u_2 do stavu x_2 atď. každá premenná x je

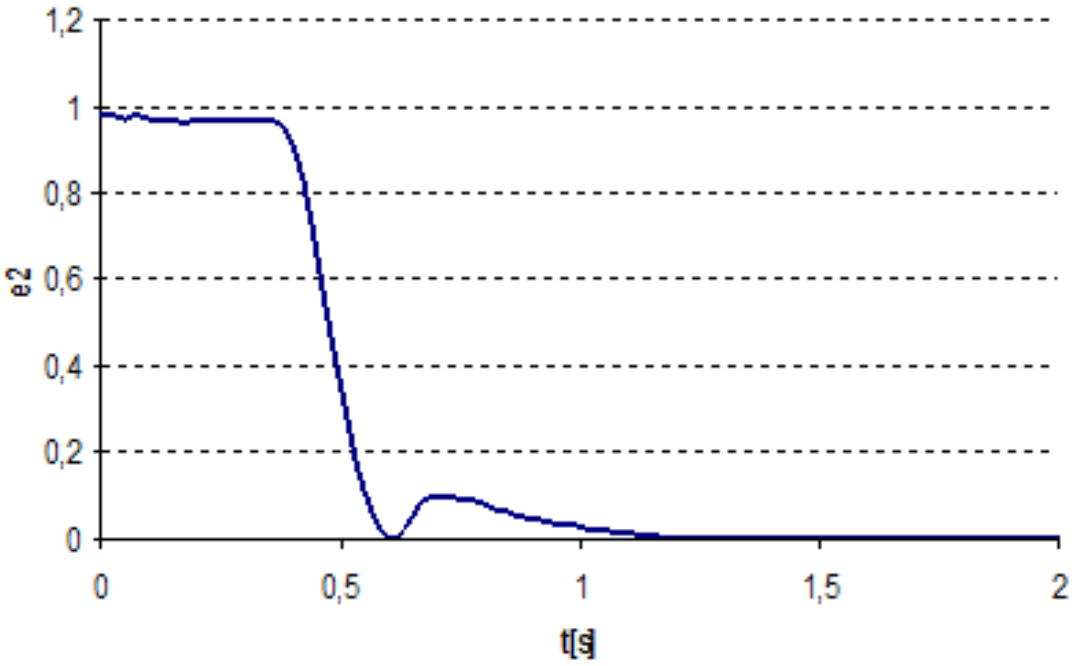
odvoditeľná z predošlého stavu a prvý stav x_0 je známy. Preto jedinou „neznámou“ ostáva sekvencia riadiacich zásahov. Jedna z otázok by mohla byť, prečo sa využíva kvadratické kritérium. Ako odpoveď je možné použiť príklad kvality riadenia. Pri hodnotení kvality riadenia sa používajú integrálne kritéria. Existuje jednoduché kritérium, ktoré spraví integrál pod krivkou regulačnej odchýlky. Nevýhodou tohto je, že ak dochádza k preregulovaniu a regulačná odchýlka je záporná, veľkosť plochy je zmenšovaná o tie časti, ktoré sú záporné. Toto sa rieši absolútnym integrálnym kritériom, ktoré spraví zo zápornej regulačnej odchýlky kladnú a teda plocha pod krivkou sa zväčšuje aj pri preregulovaní. Navyše existuje kvadratické kritérium, ktoré okrem toho, že odstraňuje problém so zápornou regulačnou odchýlkou navyše viac penalizuje hodnoty odchýlky väčšie ako 1. Inými slovami, ak je hodnota viac ako 1 o to horšiu kvalitu regulácie bude toto kritérium indikovať. Problém so zápornou regulačnou odchýlkou je znázornený na obrázkoch 2, 3 a 4.



Obrázok 2: Hodnota regulačnej odchýlky v čase

Rovnako ako pri minimalizácii regulačnej odchýlky, tak aj pri minimalizácii riadiaceho zásahu je kvadratické kritérium najlepším ukazovateľom.
Pre porozumenie ďalších vzťahov si je potrebné uvedomiť, že: $x(t+k) = x_{(t+k)}$

- Pre vzorku k = 0 (aktuálny stav systému): x(t)= x_t,



Obrázok 3: Časová závislosť kvadrátu regulačnej odchýlky

- pre vzorku k = 1 x(t+1)=$x_{(t+1)}$,
- atď.

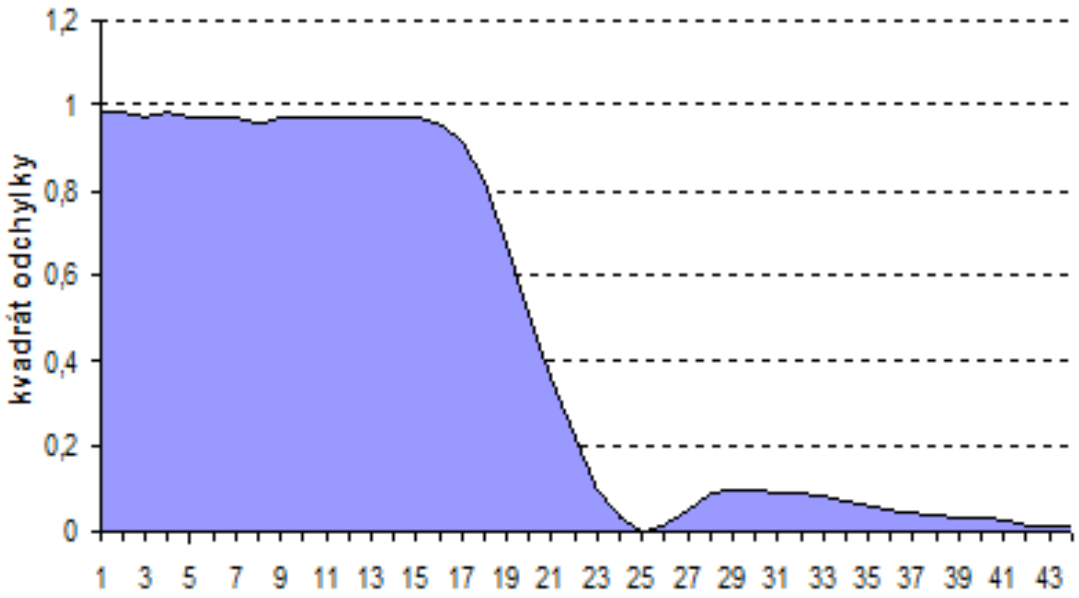
Kvadratické kritérium pre konečný horizont predikcie dĺžky N je:

$$J(x(t), u(t), \ldots, u(N-1)) = \frac{1}{2}\sum_{k=0}^{N-1}[x_k^T Q x_k + u_k^T R u_k] + \frac{1}{2}x_N^T Q_N x_N \tag{3}$$

kde:

$$\begin{aligned} x_(k+1) &= Ax_k + Bu_k \\ x_0 &= x(t), \\ Q = Q^T \succcurlyeq 0, Q_N = Q_N^T \succcurlyeq 0, R = R^T &\succ 0 \\ Q_N &\in R^{n\times n} \\ R &\in R^{m\times m} \end{aligned} \tag{4}$$

Matice, Q, Q_N a R sú nazývané váhové matice a spolu s horizontom predikcie N sú nazývané parametrami na ladenie prediktívneho regulátora. Nájdenie optimálneho riadenia, ktoré by minimalizovalo kvadratické kritérium predstavuje dynamickú optimalizáciu a má



Obrázok 4: Plocha kvadrátu regulačnej odchýlky

v tomto prípade aj analytické riešenie: [11]

$$\begin{aligned} x_{t+k} &= A^k x_0 + \sum_{i=0}^{k-1} A^i B u_{k-1-i} \\ u_{t,N} &= [u_t^T, \ldots, u_{t+N-1}^T] \end{aligned} \tag{5}$$

Vyjadrenie predikovaného stavu je potom:

$$[x_{t+1}^T, \ldots, x_{t+N}^T]^T = V x_0 + T u_{t,N} \tag{6}$$

Táto rovnica je jedna z najdôležitejších pri pochopení fungovania on-line prediktívneho algoritmu. Preto je vhodné ukázať ako matica V a T vyzerajú pre konkrétny jednoduchý systém s horizontom predikcie N = 3 zadaný v stavovom priestore:

$$\begin{aligned} A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0,5 \end{bmatrix} \\ C = \begin{bmatrix} 0 & 1 \end{bmatrix}, D = 0 \\ x_0 = \begin{bmatrix} 4 \\ 5 \end{bmatrix} = \begin{bmatrix} x_{01} \\ x_{02} \end{bmatrix} \end{aligned} \tag{7}$$

Matice V a T budú vyzerať:

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

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0,5 & 0 & 0 \\ 1 & 1 & 0 \\ 1,5 & 0,5 & 0 \\ 1 & 1 & 1 \\ 2,5 & 1,5 & 0,5 \end{bmatrix} = \begin{bmatrix} A^0B & 0 & 0 \\ A^1B & A^0B & 0 \\ A^2B & A^1B & A^0B \end{bmatrix} \tag{8}$$

Výsledný vzťah:

$$\left[\begin{bmatrix} x_{11} \\ x_{12} \end{bmatrix}, \begin{bmatrix} x_{21} \\ x_{22} \end{bmatrix}, \begin{bmatrix} x_{31} \\ x_{32} \end{bmatrix} \right]^T = \begin{bmatrix} A^1 \\ A^2 \\ A^3 \end{bmatrix} \begin{bmatrix} x_{01} \\ x_{02} \end{bmatrix} + \begin{bmatrix} A^0B & 0 & 0 \\ A^1B & A^0B & 0 \\ A^2B & A^1B & A^0B \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{9}$$

Dôležitý krok pri hľadaní optimálneho riadenia je zo vzťahu 3 „odstrániť“ sumu. Finálny vzťah vyzerá:

$$J\left(x\left(t\right), U_t\right) = \frac{1}{2} {u_{t,N}}^T H u_{t,N} + x^T\left(t\right) F U_t + \frac{1}{2} x^T\left(t\right) \mathrm{Yx}\left(t\right), \tag{10}$$

Matica $H \in R^{(m \bullet N \times m \bullet N)}$, $F \in R^{(n \times m \bullet N)}$, $Y \in R^{(n \times n)}$. Ak sa symbol $\bigotimes$ označí ako kroneckerovo násobenie matíc a $I_j \in R^{(j \times j)}$, jednotková matica s príslušnou dimenziou, tak platí:

$$\begin{gathered} H = T^T \tilde{Q} T + I_N \bigotimes R, \quad F = V^T \tilde{Q} T, \quad Y = V^T \tilde{Q} V \\ \tilde{Q} = \begin{bmatrix} I_{N-1} \bigotimes Q & 0 \\ 0 & Q_N \end{bmatrix} \end{gathered} \tag{11}$$

Pre systém zo vzťahu 7 a váhy $Q = Q_N = I_2,\ R = 0,1$ budú jednotlivé matice

vyzerať:

$$H = \begin{bmatrix} 11,85 & 6,5 & 2,25 \\ 6,5 & 4,6 & 1,75 \\ 2,25 & 1,75 & 1,35 \end{bmatrix}, F = \begin{bmatrix} 14 & 7,5 & 2,5 \\ 4,5 & 2 & 0,5 \end{bmatrix}, \quad Y = \begin{bmatrix} 17 & 6 \\ 6 & 3 \end{bmatrix} \tag{12}$$

Optimálnu riadiacu sekvenciu je možné dostať minimalizáciou kvadratickej formy 10:

$$u^*_{t,N}\left(x\left(t\right)\right) = \arg\min_{u_t,N}\left\{J(x\left(t\right), u_{t,N}\ \right\} = -H^{-1}F^T x(t) \tag{13}$$

Optimálna hodnota kritéria je:

$$J^*\left(x\left(t\right)\right) = \min_{u_t,N}\left\{J(x\left(t\right), u_{t,N}\ \right\} = \frac{1}{2}x^T(t)(Y - FH^{-1}F^T)x(t) \tag{14}$$

Na to aby sa zabezpečila spätná väzba, je potrebné v každom kroku použiť iba prvú hodnotu zo sekvencie riadiacich zásahov a potom zmerať stav a na základe tej hodnoty znovu vypočítať optimálnu sekvenciu riadiacich zásahov. Bez merania stavu, by to bola regulácia v otvorenom regulačnom obvode. Meranie stavu a jeho použitie pri výpočte nasledujúceho riadiaceho zásahu v každom kroku zabezpečí reguláciu v uzavretom regulačnom obvode.

1.1.2 On-line MPC s obmedzením

Doteraz sa reguloval systém, v ktorom nebolo žiadne obmedzenie. V realite ich však býva mnoho. Pokiaľ sa pri návrhu regulátora začnú brať do úvahy rovnice 2, bude ich treba zapracovať do výpočtu. Pri návrhu prediktívneho regulátora s obmedzením sa využíva kvadratické programovanie. Úloha kvadratického programovania je úlohou nelineárneho programovania, v ktorej sústava ohraničení je lineárna a účelová funkcia je kvadratická. Všeobecná formulácia kvadratickej úlohy je:

$$\begin{aligned} f\left(x\right) &= \min_x \left\{ x^T H x + F^T \mathrm{x} \right\} \\ x \in D &= \{\mathrm{x}|\ \ Lx \leq m_c,\ x \geq 0\} \end{aligned} \tag{15}$$

Pre návrh regulátora je premenná, ktorej minimum sa hľadá, sekvencia riadiacich zásahov $u^*_{t,N}$. Rovnice pre prediktívny regulátor s obmedzením následne vyzerajú:

$$J\left(x\left(t\right),\ u\left(t\right),\ldots,\ u\left(N-1\right)\right)=\frac{1}{2}\sum_{k=0}^{N-1}\left[x_k^TQx_k+u_k^TRu_k\right]+\ \frac{1}{2}x_N^TQ_Nx_N \tag{16}$$

kde:

$$\begin{aligned} x_{k+1}&=Ax_k+Bu_k,\\ x_0&=x\left(t\right),\\ E_cx_k+G_cu_k&\leq m_c,\quad k=1\ldots N\\ Q=Q^T\succcurlyeq 0,\ Q_N=Q_N^T\succcurlyeq 0,\ R&=R^T\succ 0 \end{aligned} \tag{17}$$

Aby bolo možné sústavu rovníc 16 a 17 zapracovať do kvadratického programovania, previesť do maticového tvaru. Prvá časť rovnice ostáva nezmenená, pridá sa k nej druhá časť:

$$\begin{aligned} J\left(x\left(t\right),U_t\right)=\ \frac{1}{2}{u_{t,N}}^THu_{t,N}+x^T\left(t\right)FU_t+\frac{1}{2}x^T\left(t\right)\mathrm{Yx}\left(t\right),\\ Gu_{t,N}\leq w+Ex(t) \end{aligned} \tag{18}$$

Kde matice H, F, Y sú určené rovnako ako vo vzťahu 11 a matice G, E, a vektor w majú tvar:

$$G=\begin{bmatrix}-I_N\\ I_N\\ -(I_N\bigotimes CT)\\ (I_N\bigotimes CT)\\ \vdots\end{bmatrix},\ E=\begin{bmatrix}0\\ 0\\ -(I_N\bigotimes CV)\\ (I_N\bigotimes CV)\\ \vdots\end{bmatrix},\ w=\begin{bmatrix}-(1_N\bigotimes u_{\min})\\ (1_N\bigotimes u_{\max})\\ -(1_N\bigotimes y_{\min})\\ (1_N\bigotimes y_{\max})\\ \vdots\end{bmatrix} \tag{19}$$

V rovnici 19 sú znázornené základné systémové obmedzenia. Môže existovať ďaleko viac. 1_N predstavuje jednotkový vektor o veľkosti N. Prvé dva riadky matice G, $G_{1,2}\in R^{N\times N}$, E, $E_{1,2}\in R^{N\times n}$ a vektor W, $W_{1,2}\in R^N$ v (19) predstavujú obmedzenia vstupu. Druhé dva riadky matice G, $G_{3,4}\in R^{N\times N}$, E, $E_{3,4}\in R^{N\times n}$ a vektor W, $W_{3,4}\in R^N$ v 19 predstavujú obmedzenia výstupu. Ďalšie obmedzenia, by museli nadobúdať rovnaké rozmery ako predošlé riadky príslušných matíc a vektora.

Ak sa pridajú do systému 7 obmedzenia na vstup a výstup:

$$-1\leq u\left(t\right)\leq 1,\quad -10\leq y\left(t\right)\leq\ 10 \tag{20}$$

matice G, E a vektor w budú vyzerajú:

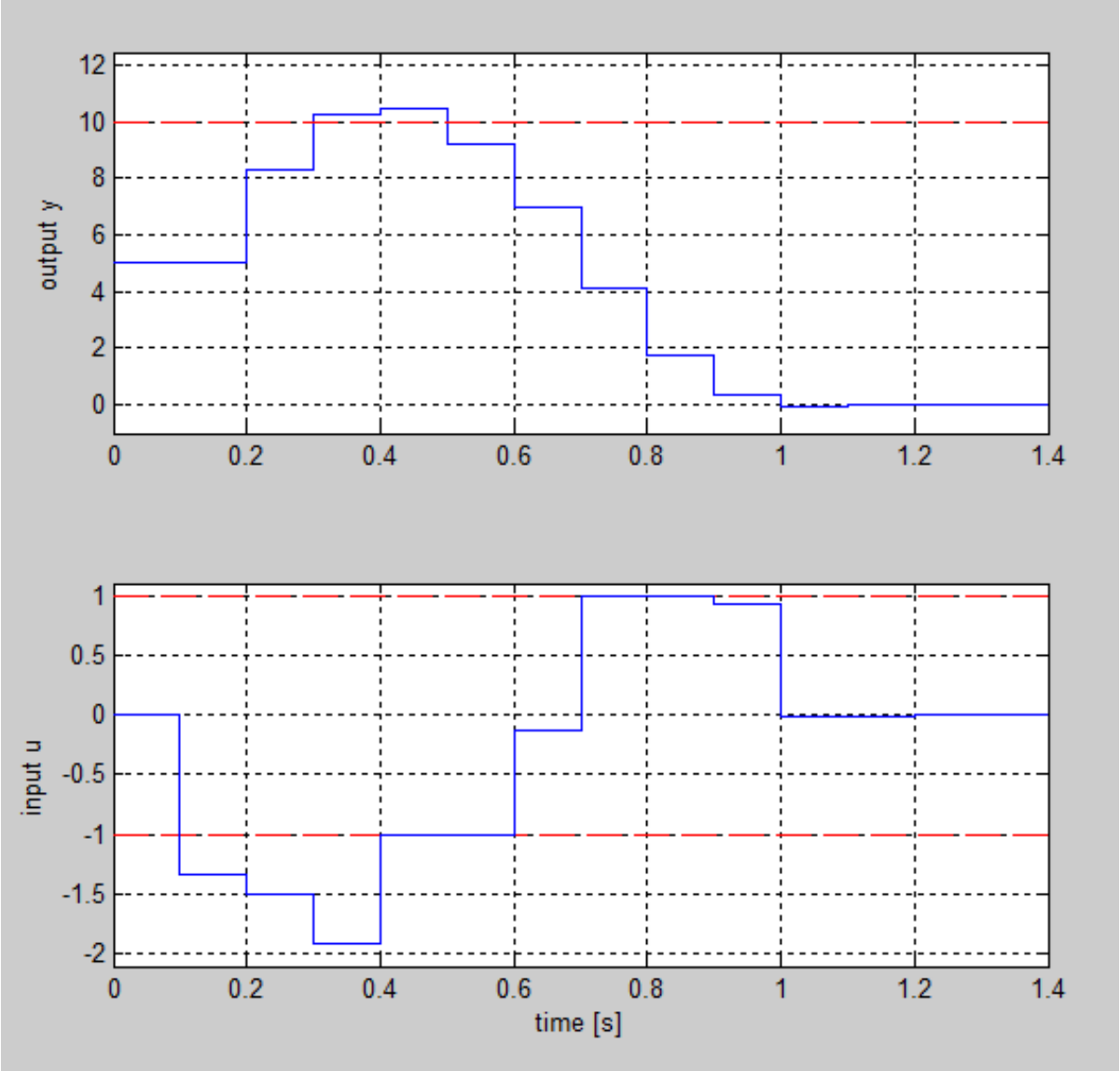
$$G = \begin{bmatrix} -I_3 \\ I_3 \\ -(I_3 \bigotimes CT) \\ (I_3 \bigotimes CT) \end{bmatrix}, \; E = \begin{bmatrix} 0 \\ 0 \\ -(I_3 \bigotimes CV) \\ (I_3 \bigotimes CV) \\ \vdots \end{bmatrix}, \; m_c = \begin{bmatrix} -(1_3 \bigotimes u_{\min}) \\ (1_3 \bigotimes u_{\max}) \\ -(1_3 \bigotimes y_{\min}) \\ (1_3 \bigotimes y_{\max}) \end{bmatrix} \tag{21}$$

$$G = \begin{bmatrix} -\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ -\begin{pmatrix} 0,5 & 0 & 0 \\ 1,5 & 0,5 & 0 \\ 2,5 & 1,5 & 0,5 \end{pmatrix} \\ \begin{pmatrix} 0,5 & 0 & 0 \\ 1,5 & 0,5 & 0 \\ 2,5 & 1,5 & 0,5 \end{pmatrix} \end{bmatrix}, \; E = \begin{bmatrix} -\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \\ -\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} \\ \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} \end{bmatrix}, \; w = \begin{bmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ \begin{pmatrix} 10 \\ 10 \end{pmatrix} \\ \begin{pmatrix} 10 \\ 10 \end{pmatrix} \end{bmatrix} \tag{22}$$

Vznikne sústava lineárnych rovníc, ktoré tvoria obmedzenia pre kvadratický problém.

$$\begin{bmatrix} -\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ -\begin{pmatrix} 0,5 & 0 & 0 \\ 1,5 & 0,5 & 0 \\ 2,5 & 1,5 & 0,5 \end{pmatrix} \\ \begin{pmatrix} 0,5 & 0 & 0 \\ 1,5 & 0,5 & 0 \\ 2,5 & 1,5 & 0,5 \end{pmatrix} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \leq \begin{bmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ \begin{pmatrix} 10 \\ 10 \end{pmatrix} \\ \begin{pmatrix} 10 \\ 10 \end{pmatrix} \end{bmatrix} + \begin{bmatrix} -\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ -\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} \\ \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} \end{bmatrix} \begin{bmatrix} x_{01} \\ x_{02} \end{bmatrix} \tag{23}$$

Ak sú obmedzenia na systém príliš striktné, môže sa stať, že kvadratický problém nemá riešenie. Takáto situácia je znázornená na obrázku 5.

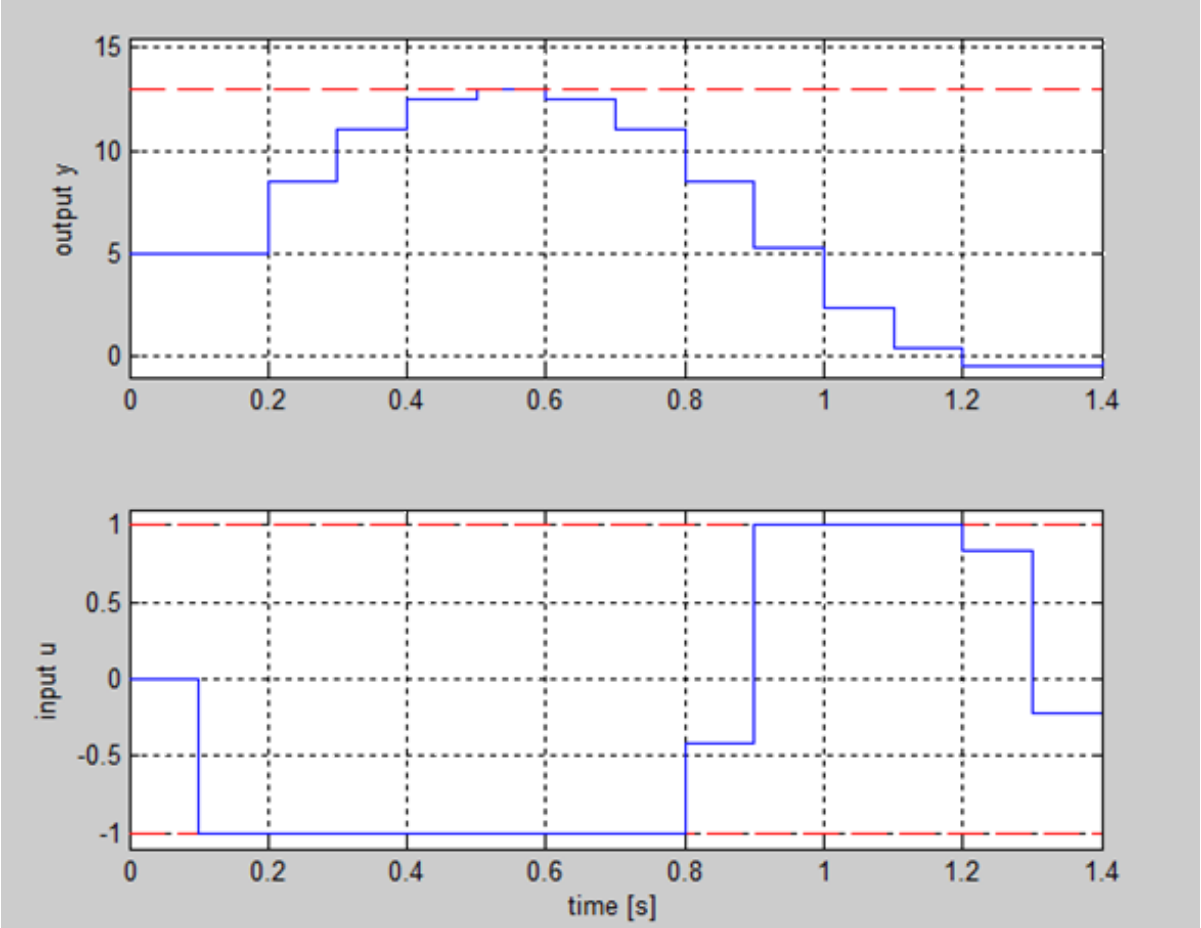


Obrázok 5: Časový priebeh výstupnej veličiny a riadiaceho zásahu pre neriešiteľný kvadratický problém. Červená čiara predstavuje obmedzenie na danú veličinu

Ak by obmedzenia na výstup boli napríklad $-13 \leq y\,(t) \leq\ 13$, kvadratický problém by bol riešiteľný a jeho riešenie pre horizont predikcie 3 je zobrazene na obrázku 6.

1.1.3 On-line MPC so sledovaním referenčného signálu

Doteraz bol riešený regulátor, ktorý reguloval hodnotu k „počiatku“ k hodnote 0 alebo nulovému vektoru. Táto kapitola obsahuje návrh MPC regulátora, ktorý bude sledovať po častiach konštantný referenčný signál r(t). Stále sa berie do úvahy systém 1. Najprv



Obrázok 6: Časový priebeh výstupnej veličiny a riadiaceho zásahu riešiteľného kvadratického problému

je potrebné nahradiť člen $x_k^T Q x_k$ v pôvodnom kritériu 17 odchýlkou

$$(z_k - r_k)^T Q_y (z_k - r_k) \tag{24}$$

kde

$$z(t) = Zy(t), \; z(t) \in R^{p_z}, \; p_z \leq p, \; r \leq m \tag{25}$$

Aby to bolo možné, je potrebné poznať predikovanú hodnotu referenčného signálu. Ta môže byť definovaná rôznymi spôsobmi jedna z možností: r(t+1)=r(t).

V ustálenom stave je potrebné, aby platila rovnosť $z_u = r_u$. Takže:

$$\begin{bmatrix} I_n - A & -B \\ \text{ZC} & 0 \end{bmatrix} \begin{bmatrix} x_u \\ u_u \end{bmatrix} = \begin{bmatrix} 0 \\ r_u \end{bmatrix} \tag{26}$$

Ak ma predchádzajúca matica plnú riadkovú hodnosť, existuje riešenie u_u a x_u.. V dôsledku nenulového ustáleného vstupu sa v praxi často používa odchýlka $u_k = u_k - u_{k-1}$. Ak sústava obsahuje integrátor, je lepšie použiť hodnotu u. [11]

Kritérium pre sledovanie referencie má tvar:

$$J\left(x\left(t\right),\ u\left(t\right),\ldots,\ u\left(N-1\right)\right) = \frac{1}{2}\sum_{k=0}^{N-1}\left[e_k^T Q_e e_k + u_k^T R u_k\right] \tag{27}$$

kde

$$e_k = y_k - r_k \tag{28}$$

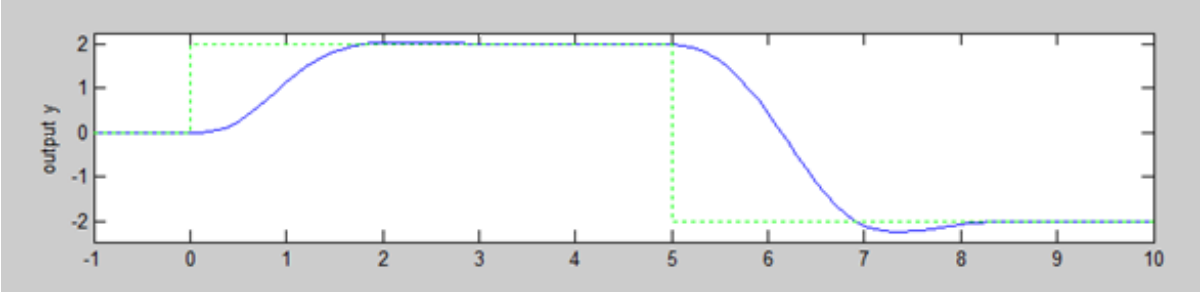
je regulačná odchýlka a

$$\Delta u_k = u_k - u_{k-1} \tag{29}$$

je optimalizovaná premenná. Systém rozšírený o históriu riadiaceho zásahu u_k a generátor referencie r_k je formulovaný:

$$\begin{bmatrix} x_{k+1} \\ u_k \\ r_{k+1} \end{bmatrix} = \begin{bmatrix} A & B & 0 \\ 0 & I_m & 0 \\ 0 & 0 & I_{n_y} \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \\ r_k \end{bmatrix} + \begin{bmatrix} B \\ I_m \\ 0 \end{bmatrix} u_k = \begin{array}{l} A_m \hat{x_k} + B_m u_k \\ e_k = \begin{bmatrix} C & 0 & -I_{n_y} \end{bmatrix} \hat{x_k} = C_m \hat{x_k} \end{array} \tag{30}$$

Po upravení algoritmu na sledovanie po častiach spojitého referenčného signálu už MPC regulátor neriadi výstupnú veličinu k 0 hodnote alebo vektoru, ale ku aktuálnej hodnote prípadne vektoru referenčného signálu v kroku k, . . . , k + N - 1, kde N predstavuje horizont predikcie. Preto referenčný signál vstupujúci do výpočtu je je buď vektor pre jednorozmerné systémy (SISO) alebo matica pre viacrozmerné systémy (MIMO). Časový priebeh výstupnej veličiny SISO systému regulovaného MPC regulátorom so sledovaním referencie je znázornený na obrázku 7.



Obrázok 7: Sledovanie referencie, časové priebehy - zelená referencia, modrá výstup systému

1.1.4 Explicitné riešenie - offline MPC

Je zrejmé, že optimálna hodnota kritéria 14 a optimálna riadiaca sekvencia 13 (aj s obmedzeniami) sú funkciou stavu x(t). Túto úlohu je možné formulovať pomocou **multiparametrického kvadratického programovania** (mp-QP), ktoré sa snaží nájsť optimálne riešenie pre všetky možné hodnoty parametra x(t) vopred. Ak sa spraví substitúcia rovnice

$$J^*\left(x\left(t\right)\right) = \min_{u_t,N} \left\{J\left(x\left(t\right), u_t\right) \middle| Gu_t \leq W + Ex(t)\right\} \tag{31}$$

pomocou

$$u_t = \tilde{u_t} - H^{-1}F^T x(t) \tag{32}$$

vznikne:

$$J^*\left(x\left(t\right)\right) = \min_{u_t,N} \left\{J\left(\tilde{u_t}, x\left(t\right)\right) = \frac{1}{2}\tilde{u_t}^T H\tilde{u_t} + \beta \middle| G\tilde{u_t} \leq W + Sx(t)\right\} \tag{33}$$

Kde $S = E + GH^{-1}F^T$ a $\beta = \frac{1}{2}x\left(t\right)^T \left(FH^{-1}F^T + Y\right) x(t)$. Ešte je potrebné zaviesť množinu indexov I= $\{1, \ldots, q\}$, ktorá zodpovedá riadkom matice G, W a S.
Kritický región CR je taká polytopická oblasť v priestore parametrov x(t), ktorá má v optimálnej hodnote $J^*\left(x\left(t\right)\right)$, $u^*\left(x\left(t\right)\right)$ aktívne rovnaké obmedzenia $A(x(t)) \subset I$. Takže platí

$$G_A\tilde{u_t}^*\left(x\left(t\right)\right) = W_A + S_A x\left(t\right) \;\; pre \; x(t) \in CR \tag{34}$$

Nech $H \succ 0$ a nech $G_A$majú lineárne nezávisle riadky, potom optimálna sekvencia $\tilde{u_t}^*(x(t))$ je jednoznačne definovaná afinnou funkciou stavu x(t) na danom kritickom regióne CR.

$$\tilde{u_t}^*(x(t)) = H^{-1}G_A^T(G_A H^{-1} G_A^T)(W_A + S_A x(t)) \tag{35}$$

Ak sa preformuluje optimalizačný problém 13 (s obmedzeniami) ako mp-QP a $H \succ 0$, potom optimálna riadiaca sekvencia $u_t{}^*(x(t)) : X_{\text{feas}} \to R^m$ je spojitá, po častiach afinná funkcia na polyedry a optimálna hodnota $J^*(x(t))$ je spojitá, konvexná a po častiach kvadratická funkcia na polyedry.

Algoritmus mp-QP najskôr spočíta riešenie (13) (s obmedzeniami) pre vhodne zvolenú počiatočnú podmienku ($x_0 \in X_{\text{feas}}$) a vytvorí sa príslušný kritický región CR_0. Potom rekurzívne prehľadá okolie a vytvára nové regióny. Výsledkom je rozdelenie oblasti X_{feas}do kritických regiónov $CR_i = \{x|\, P_i x \leq p_i\}$, nad ktorými je definovaná spojitá afinná funkcia:

$$u_t{}^*(x(t)) = F_i x(t) + G_i \tag{36}$$

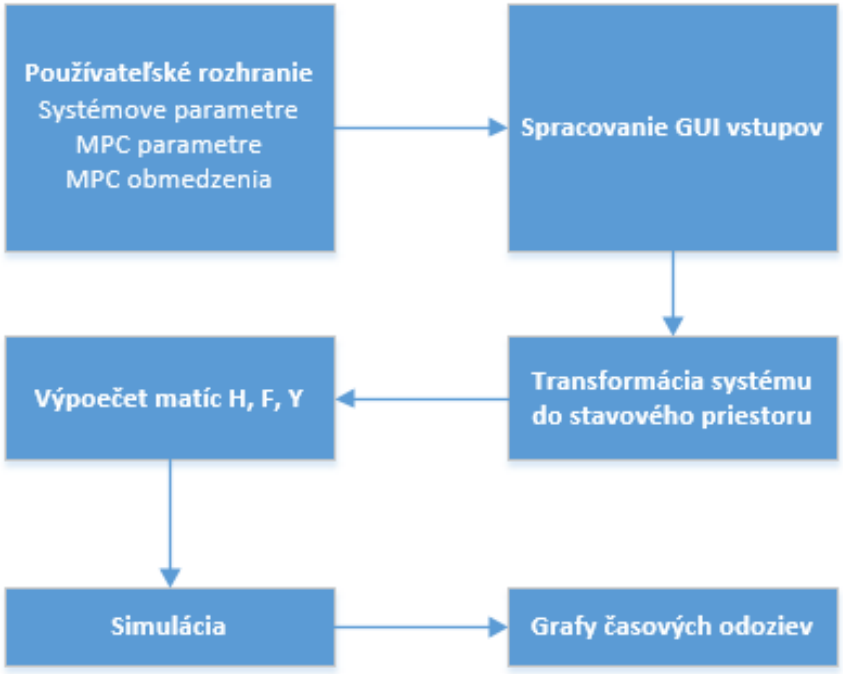
a spojitá kvadratická funkcia

$$J_t{}^*(x(t)) = x^T(t) A_i x(t) + B_i x(t) + C_i \tag{37}$$

Týmto sa presunula numerická výpočtová náročnosť optimalizácie 13 k off-line výpočtom. V priebehu riadenia stačí identifikovať región CR_i, obsahujúci aktuálny stav x(t) a aplikovať príslušný zákon riadenia . [11]
Takýmto spôsobom je možné rozdeliť výpočtovú zložitosť na dve časti. Prvá, výpočtovo zložitejšia, časť je hľadanie kritických regiónov, ktorá sa môže uskutočniť pred zavedením regulátora do behu. Tuto nie je čas kritický, pretože regulačný proces nebeží. Druhá časť je počas behu regulačného procesu. Tá predstavuje časovo nenáročnú operáciu vyhľadania kritického regiónu v pamäti a podľa neho vrátiť akčný zásah. Hlavná nevýhoda offline - explicitného riešenia je, že systém je jednorázovo daný a už počas behu nevstupuje do výpočtu na rozdiel od online riešenia, kde do každého kroku matematický model systému vstupuje a teda je možné matematický model dynamicky adaptovať, aby čo najviac zodpovedal reálnemu systému. Pri voľbe spôsobu implementácie praktickej časti pre výučbu aj neskôr v IoT prostredí, tento fakt zavážil a zvolila sa online metóda implementácie.

1.2 Popis funkčnosti On-line MPC algoritmu

Vytvorený algoritmus sa skladá skladá z viacerých modulov - funkcií naprogramovaných v prostredí Matlab a prispôsobených na fungovanie v open source verzii Octave. Programbol vytvorený pre účely využitia v pedagogickom procese grafické rozhranie bolo vytvorené pomocou komponent Matlabu - Guide, ktorý slúži práve na vytváranie grafických rozhraní. Základné komponenty, z ktorých sa program a skladá sú znázornené na obrázku 8.



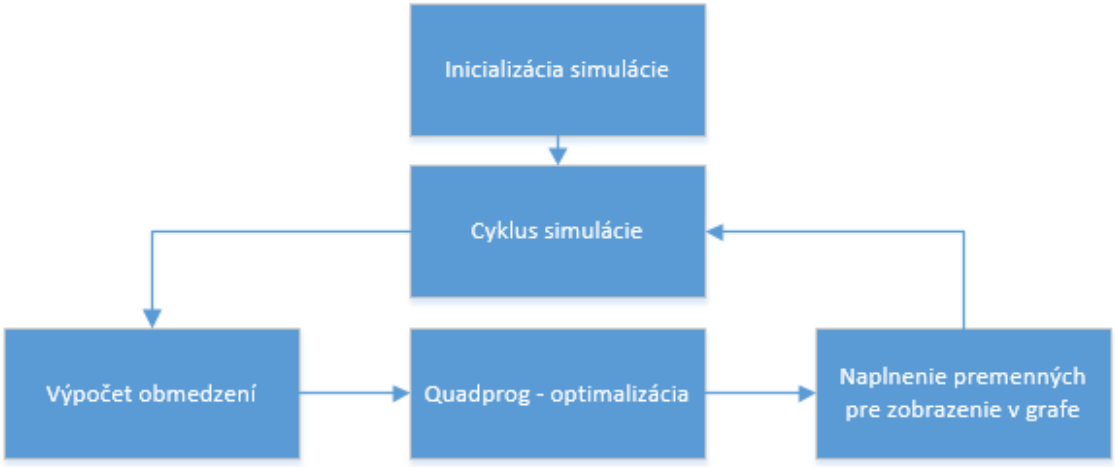
Obrázok 8: Základné komponenty

1. Používateľské rozhranie je popísane v kapitole 1.3
2. Spracovanie vstupov zabezpečí správnu konverziu reťazcov na čísla a výsledné čísla priradí do správnych premenných potrebných na vstupe do ďalšej časti.
3. MPC regulátor pre svoje fungovanie vždy potrebuje mať na vstupe matematický model systému podľa rovnice 1, ktorý sa označuje ako systém zadaný v stavovom priestore. Táto časť zabezpečí konverziu z prenosovej funkcie na stavový priestor.

4. Nasleduje výpočet matíc H, F, Y podľa rovníc 11, ktoré sú produktom matíc A, B, C z rovnice 1 a váhových matíc, Q, Q_n a R z rovnice 11, ktoré si používateľ volí.

5. Následne prebieha simulácia, ktorej komponenty sú znázornené na obrázku 9 a popísane nižšie.

6. Produktom simulácie sú dáta, ktoré vytvoria grafy časových odoziev popísaných v kapitole 1.3.



Obrázok 9: Detail komponentu simulácia

1. Počas inicializácie simulácie sa pripravuje referenčný signál, počiatočný stav systému a počet krokov simulácie.

2. Počet krokov simulácie je nastavený na podiel času simulácie a periódy vzorkovania.

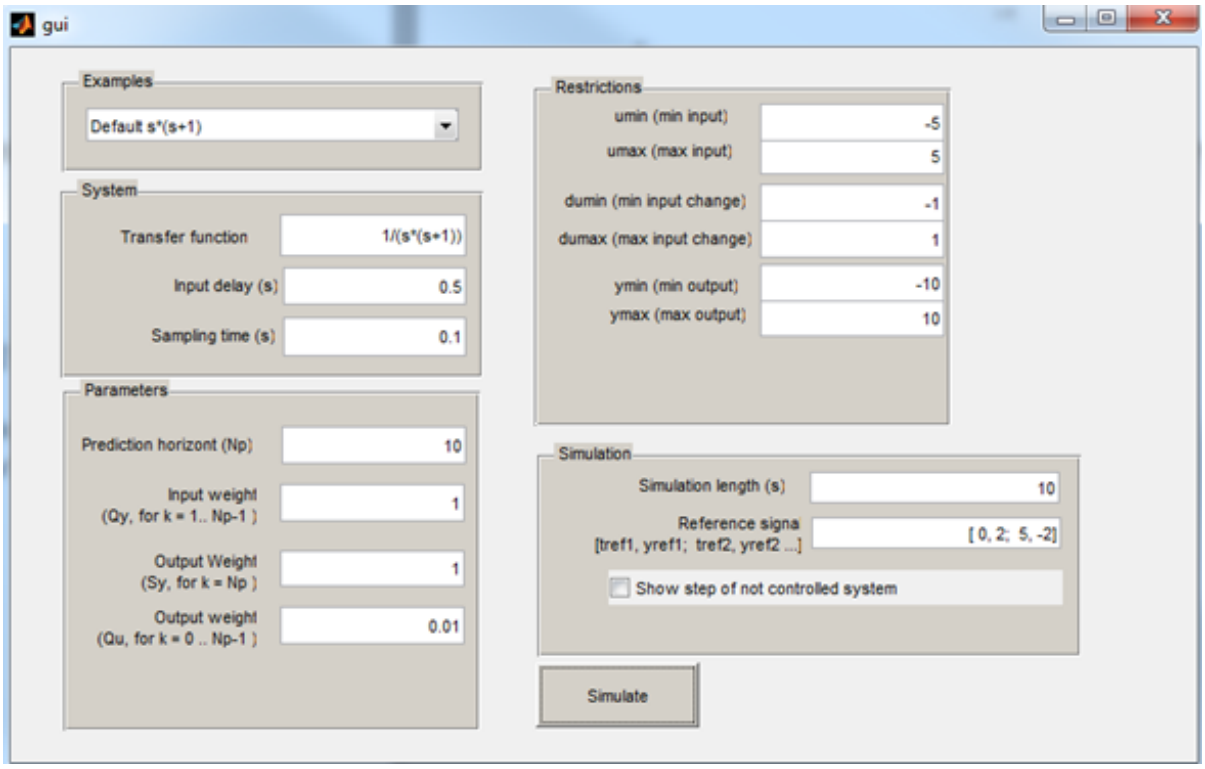
3. V prvej fáze simulačného cyklu sa prepočítajú obmedzenia systému podľa rovnice 19.

4. Následne prebehne samotná optimalizácia spustením Matlab súčasti *quadprog*, prípade Octave súčasti *qp*. Bez obmedzení by to bol výpočet podľa rovnice 13 spomenuté súčasti kvadratického programovania k tomu pridajú sústavu obmedzení.

5. Po optimalizácii sa vyberie prvý akčný zásah z vektora akčných zásahov, ktorý má dĺžku horizontu predikcie, čo je vlastne výsledok optimalizácie. Keďže v programe beží simulačný cyklus tak sa vypočíta výstup, a stav, ktoré sa zapamätajú a následne začína ďalší krok simulačného cyklu.

1.3 Overenie On-line MPC algoritmu

Po teoretickom základe a popise vytvoreného algoritmu nasleduje popis grafického rozhrania a jeho overenie. Grafické rozhranie vytvoreného programu je znázornené na obrázku 10



Obrázok 10: Grafické rozhranie k programu na testovanie MPC algoritmu.

a ponúka pre používateľa možnosti zadania:

- Systémových nastavení:
  - Automatický vyplniť určité druhy systému
  - Zadať akúkoľvek prechodovú funkciu v s alebo z oblasti systému, ktorý má byť riadený
  - Zadať dopravné oneskorenie systému
  - periódu vzorkovania
- Ladiace parametre
  - váha vstupu

  - váha výstupu pre budúce hodnoty v kroku k=0 . . . N-1, N je horizont predikcie.
  - váha výstupu pre budúcu hodnotu v kroku k=N, kde N je horizont predikcie

- Obmedzenia systému
  - minimálny vstup
  - maximálny vstup
  - minimálna zmena vstupu
  - maximálna zmena vstupu
  - minimálny výstup systému
  - maximálny výstup systému

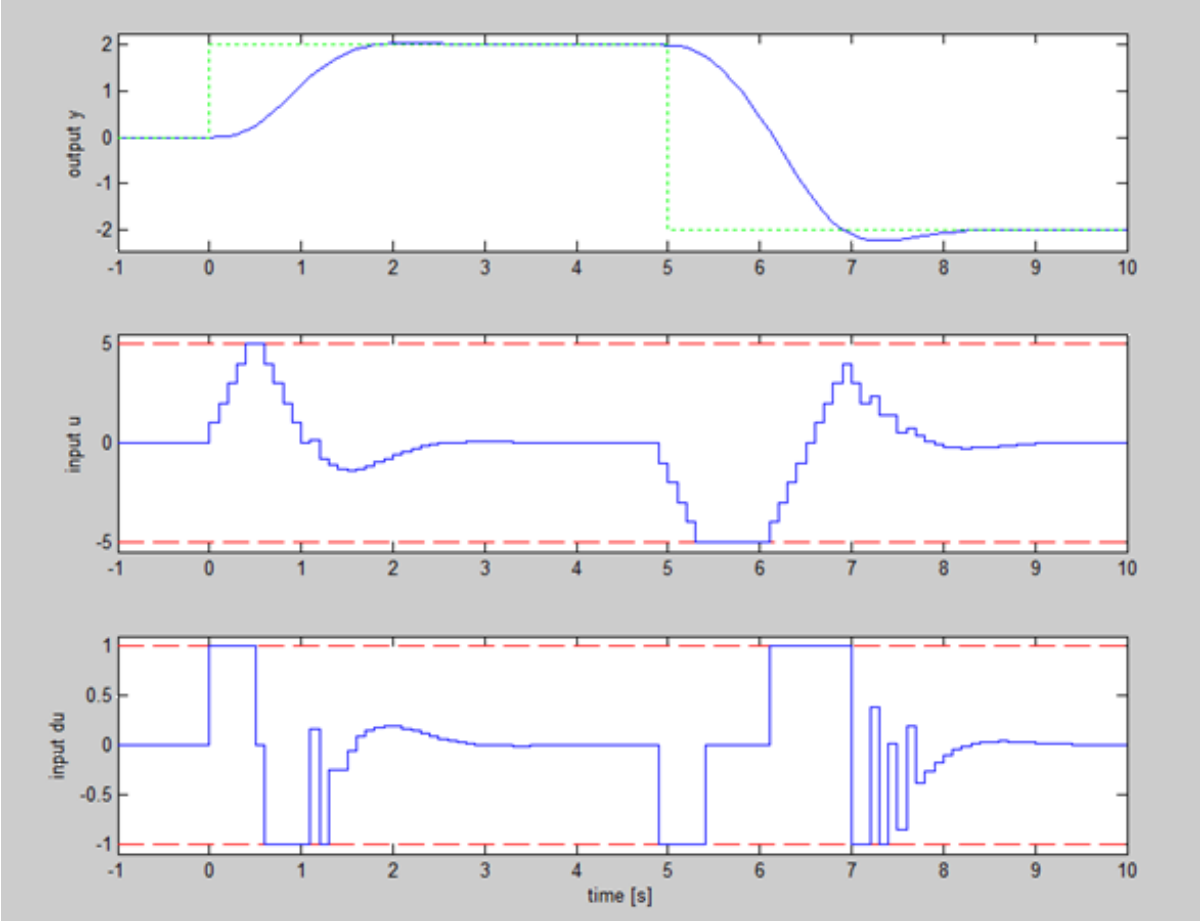
- Parametre simulácie
  - čas simulácie
  - referenčný signál zadávaný vo forme vektor dvojice hodnôt, kde prvá identifikuje čas, kedy ma zmena nastať a druhá hodnota veľkosti skoku.

Výstup z grafického rozhrania po ponechaní prednastavených hodnôt sú grafy zobrazené na obrázku 11.

V prvom grafe obrázku 11 je znázornený časový priebeh sledovania referenčnej hodnoty výstupnou veličinou. Zelenou farbou je referenčná veličina a výstupna veličina modrou. V druhom grafe obrázku 11 je znázornený riadiaci zásah systému modrou farbou a obmedzenia riadiaceho zásahu červenou farbou. V treťom grafe obrázku 11 je znázornená zmena riadiaceho zásahu modrou farbou a obmedzenia červenou. Hodnotenie kvality regulácie priamymi ukazovateľmi [7]

- Doba regulácie:
  - Pre zmenu v čase 0s (zmena 1.) z hodnoty 0 na 2 je doba regulácie 2 sekundy
  - Pre zmenu v čase 5s (zmena 2.) z hodnoty 2 na -2 je doba regulácie 3 sekundy

- Preregulovanie:
  - Pre zmenu 1. došlo k minimálnemu preregulovaniu.
  - Pre zmenu 2. je preregulovanie badateľné.



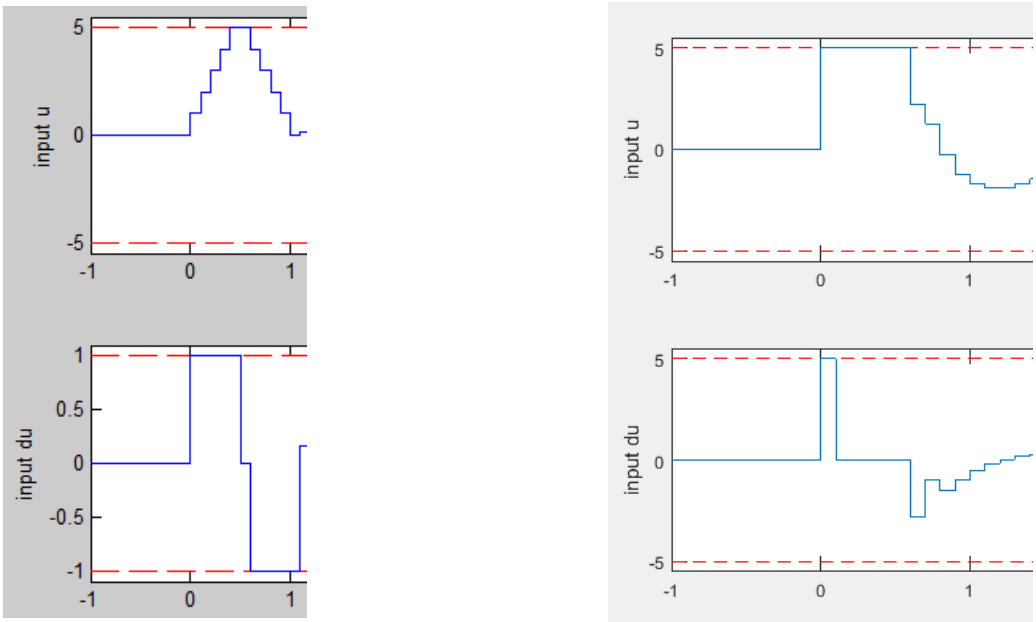
Obrázok 11: Časové priebehy výstupu systému, riadiaceho zásahu a zmeny riadiaceho zázsahu.

- Regulačná odchýlka:
  - Regulačná odchýlka je pre oba prípady 0.

Nie je tu porovnanie voči iným regulátorom, pretože to nie je zámerom tejto kapitoly. Kapitola slúži na demonštráciu funkčnosti predchádzajúcej teórie. Čo je však dôležité si na tomto mieste zdôrazniť je znázornenie ako vplývajú obmedzenia na riadenie systému.

- **Obmedzenie veľkosti akčného zásahu** je možné pozorovať v strednom grafe obrázku 11 pri zmene 2. v čase 5.3 sekundy, kedy vidieť ako systém narazil na obmedzenie akčného zásahu a na tejto hodnote ostane až do času 6.1 sekundy. Ak by obmedzenie nebolo samozrejme by bola doba regulácie kratšia avšak akčný zásah bez obmedzení vo väčšine prípadov nezodpovedá realite.
- **Obmedzenie veľkosti zmeny akčného zásahu** je opäť možné pozorovať v strednom grafe obrázku 11 pri obidvoch zmenách referenčného signálu. Zvýraznené to

je v obrázku 12a. Riadiaci zásah v tvare „schodov“ je dôsledkom tohto obmedzenia zmeny akčného zásahu. Z obrázku 12a je možné vyčítať periódu vzorkovania 0.1 sekundy, keďže je v grafe za 1 sekundu 10 zmien akčného zásahu. Prípad, že obmedzenie zmeny akčného zásahu je nastavené na hodnotu 5 (rovnaká ako obmedzenie akčného zásahu) je zobrazené na obrázku 12b.



(a) Zapnuté obmedzenie

(b) Vypnuté obmedzenie.

Obrázok 12: Ukážka vplyvu obmedzenia zmeny riadiaceho zásahu na časový priebeh riadiaceho zásahu.

Na základe týchto experimentov sa potvrdzujú výhody MPC regulátora, ktoré boli spomenuté v úvode ohľadne jednoduchosti zavedenia obmedzení. V tomto má MPC bližšie spojenie s reálnym systémom ako iné regulátory napr. PID, kde sa obmedzenie akčného zásahu musí špeciálne riešiť.

Ďalší nástroj na overenie MPC algoritmu je voľne stiahnuteľný a použiteľný v prostredí Matlab. Názov je MPT toolbox. Je to dielo inštitútu pre automatizáciu vo Švajčiarsku. Tento program je jeden z najpoužívanejších v prostredí Matlab. Nasleduje overenie algoritmu na reálnom systéme z praxe, ktorý je popísaný v nasledujúcej podkapitole. Následne je identifikovaný matematický model systému a overenie funkčnosti MPC princípov. Treba spomenúť, že overenie je stále prostredníctvom simulácie v prostredí Matlab.

1.3.1 Popis systému riadenia plynovej turbíny

Turbína s výkonom 1.5MW obsahuje tri hlavné časti, menovite, kompresor, spaľovaciu komorou a vysokotlakovú turbínu. V rokoch 1950 bolo priemyselné využitie turbín významne rozšírené, kvôli jej nespočetným výhodám. Napríklad neprítomnosť častí, ktoré by sa o seba odierali, nízka spotreba palív a vysoká operačná spoľahlivosť. Prvá časť turbíny zahŕňa stláčanie vzduchu na poskytnutie vysokého pomeru tlaku medzi turbínou a kompresorom, takže sa vzduch rozpína do turbíny. Zvyšovanie teploty vzduchu spaľovaním paliva spôsobuje väčšie rozpínanie horúceho vzduchu v turbíne, poskytujúc tak potrebný výstupný výkon. Pre rôzne prietoky vzduchu je limitovaný pomer vzduchu, ktorý môže byť dodaný, čo je označované ako pomer palivo/vzduch. Tento faktor obmedzuje výstupný výkon, ktorý môže byť dosiahnutý. Maximálny pomer palivo/vzduch je určený pracovnou teplotou lopatiek turbíny, ktoré sú vysoko stláčané. Zaznamenanie možnej poruchy lopatiek je dôležitý problém pri monitoringu a detekcii chýb. Primárna požiadavka na riadenie je výstupný výkon, ale neexistuje žiadny vhodný spôsob merania výkonu. Premenné súvisiace s výkonom sú riadené prostredníctvom riadenia generátora rýchlosti N_g, moduláciou prívodu paliva, kde N_g je funkciou výkonu generátora. Riadiaci zásah určuje množstvo paliva dodávaného do motora, čo je funkciou uhla otvorenia klapky θ_v () . Parametre systému a regulátora: rýchlosť motora leží medzi 0 a 30 000 rpm (=500rps) táto premenná môže byť považovaná za známu a reprezentuje od 0 po 1.5MW. Vstup do systému je náklon plynovej klapky 0-60°, čo je ekvivalent toku paliva 0-625kg/h. Riadiaci rozsah výkonu je od 0 po plný výkon. Od 17 001-27 000 rpm (283,35-450rps) je považovaný za stabilný stav rýchlosti generátora.[12]

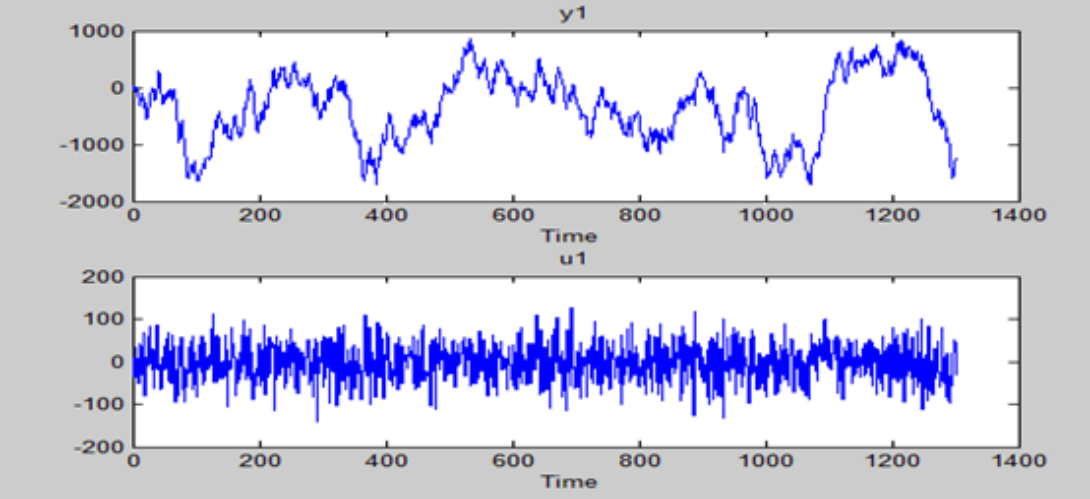
1.3.2 Identifikácia a riadenie systému plynovej turbíny

Vstupné dáta na identifikáciu systému sú zobrazené na obrázku 13.

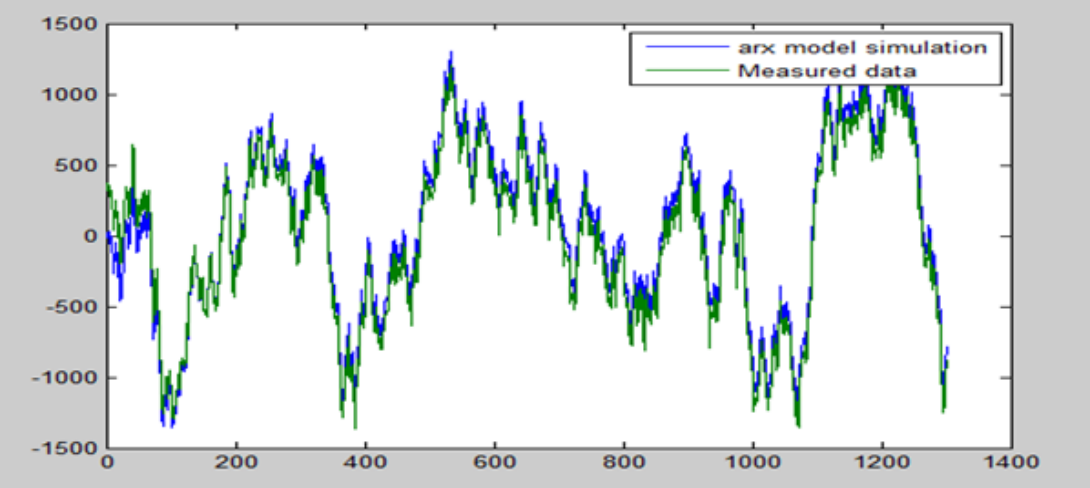
Premenná y_1 reprezentuje výstup - otáčky motora a u_1 vstup systému - natočenie klapky. Na identifikáciu systému bol použitý Matlab nástroj *ident*. Pri zisťovaní matematického modelu bolo vyskúšaných viacero metód. Najlepšie výsledky - percento zhody nameraných a simulovaných dát dal arx model. Porovnanie nameraných a simulovaných dát sú na obrázku 14.

Os x predstavuje čas v sekundách a y predstavuje výstup systému. Pomocou modelu arx sa získala nasledovná prenosová funkcia

$$Gp(z) = \frac{Y(z)}{U(z)} \tag{38}$$



Obrázok 13: Časový priebeh nameraných údajov výstup a vstup systému.



Obrázok 14: Porovnanie simulovaných a nameraných údajov

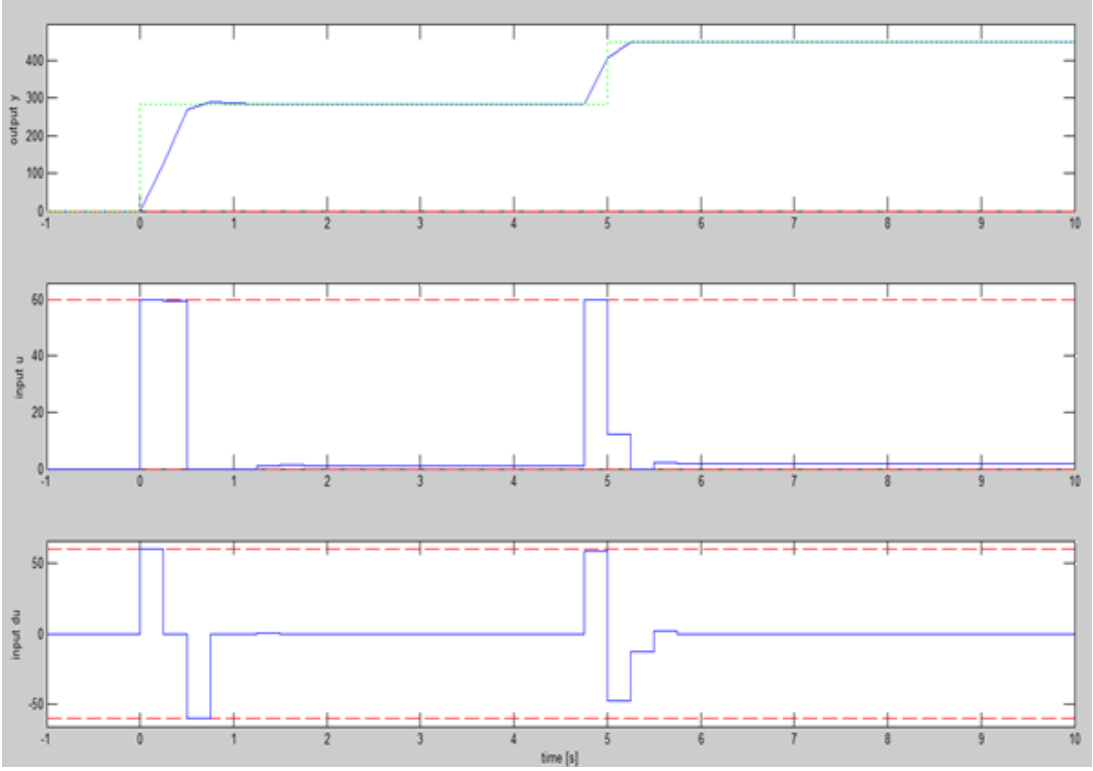
kde

$$\begin{aligned} \mathrm{Y}(z) &= 2.085z^{-1} + 0.3692z^{-2} \\ U(z) &= 1 - 0.9913z^{-1} + 1.287 \times 10^{-3}z^{-2} \end{aligned} \tag{39}$$

Perióda vzorkovania je $T_s = 0.25$.

Navrhnutý algoritmus bol otestovaný na reálnom systéme s prenosovou funkciou 39. Parametre simulácie sú zobrazene na obrázku 16. Časové odozvy systému s MPC regulátorom sú na obrázku 15. Konkrétne časová odozva výstupu, v tomto prípade výkon motora, meraný v rps (otáčky za sekundu) je zobrazený vo vrchnom grafe. Vstup systému, uhol náklonu plynovej klapky meraný v stupňoch je v strednom grafe a zmena vstupu v spodnom grafe. Všetky spomenuté veličiny majú v grafe modrú farbu. Ze-

lená farba vo vrchnom grafe predstavuje referenčný signál a červené čiarkované čiary sú obmedzenia.



Obrázok 15: Časové odozvy rýchlosti motora (v jednotkách rps) s MPC regulátorom

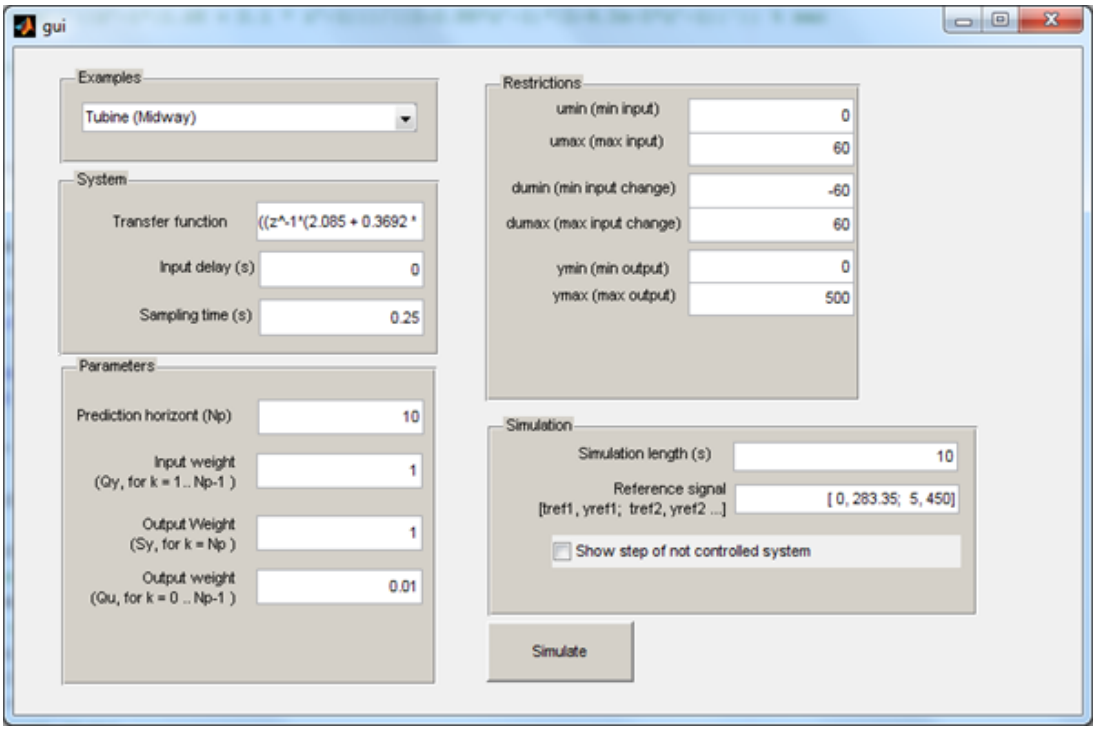
renosová funkcia 39 sa zadala do grafického rozhrania do pola „transfer function“. Horizont predikcie bol nastavený na 10 vzoriek dopredu. Zo simulácii vyplýva, že je to dostatočné a efektívne, berúc do úvahy odozvu systému a kvalitu riadenia. Váhy boli nastavené na prednastavené hodnoty. Obmedzenia algoritmu sú

- Na vstup 0-60 stupňov.
- Zmena vstupu -60 – 60 stupňov.
- Výstup systému 0 – 500 rps (30 000 rpm)

Referenčný signál je nastavený na hodnotu 283.35 rps v čase 0. Po 5 sekundách je nastavený na hodnotu 450 rps.

Z výsledkov je možné vidieť výhody prediktívneho riadenia. Keďže prediktívny regulátor je založený na optimalizácii, je možné vidieť minimálne akčné zásahy, čo môže viesť k úspore spotreby.[13]

Týmto končí simulačné overovanie algoritmu MPC regulátora vytvoreného na základe



Obrázok 16: Parametre simulácie

teoretických princípov popísaných v úvodných kapitolách MPC regulátora a nasleduje druhá časť práce, ktorá pripraví podklady pre praktický experiment.

2 Softvérová Architektúra

Téma softvérovej architektúry (Software architectrue) zahŕňa veľa súčastí. Práca sa zameriava len na tie časti, ktoré vedú do problematiky internetu vecí a teda nepojednáva o všetkých aspektoch tejto problematiky. Nasleduje niekoľko definícií a vymenovanie typov aplikácii.

Softvérová architektúra je proces definovania štrukturovaného riešenia, ktoré spĺňa všetky technické a operačné požiadavky, zatiaľ čo optimalizuje bežné kvalitatívne vlastnosti ako je výkonnosť, bezpečnosť a ovládateľnosť. Zahŕňa rad rozhodnutí, ktoré sú založené na viacerých faktoroch a každé z týchto rozhodnutí môže mať značný dopad na kvalitu, výkonnosť, udržiavanie a celkový úspech aplikácie.

Philippe Kruchten, Grady Booch, Kurt Bittner a Rich Reitman odvodili a vylepšili definíciu architektúry, ktorá vychádza z práce Mary Shaw a David Garlan (Shaw and Garlan 1996). Ich definícia je:

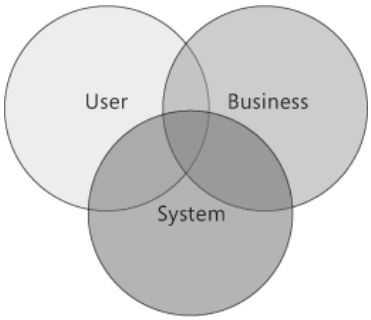
Softvérová architektúra zahŕňa sadu dôležitých rozhodnutí o usporiadaní softvérového systému vrátane voľby stavebných prvkov a ich rozhraní, pomocou ktorých je systém zložený. Rozhodnutie o voľbe správania, ktoré je definované spoluprácou medzi uvedenými prvkami. Rozhodnutie o spájaní týchto štrukturálnych a behaviorálnych elementov do rozsiahlejších subsystémov a voľba architektonického štýlu, ktorý vedie toto usporiadanie. Softvérová architektúra tiež zahŕňa starosť o funkcionalitu, použiteľnosť, pružnosť, výkonnosť, možnosť znovu použitia, zrozumiteľnosť, kompromisy na ekonomické a technologické obmedzenia a tiež estetickosť. [2]

Dôležité si je uvedomiť, že softvérová architektúra je prostriedok na vytvorenie softvérového systému avšak softvérový systém je stále len prostriedok na dosiahnutie určitého cieľa.

Preto by systémy mali byť navrhované s prihliadaním na **používateľa**, **systém** (IT infraštruktúra) a **biznis ciele**, ako je zobrazené na obrázku 17. Pre každú z týchto oblastí by mal byť načrtnutý kľúčový scenár a identifikované dôležité kvalitatívne vlastnosti (napríklad spoľahlivosť a škálovateľnosť) a kľúčové oblasti uspokojenia alebo neuspokojenia. Vytvoriť metriky a prihliadať na ne pri meraní úspechu v každej z oblasti, všade kde je to možné. [2]

2.1 Základne typy architektúr

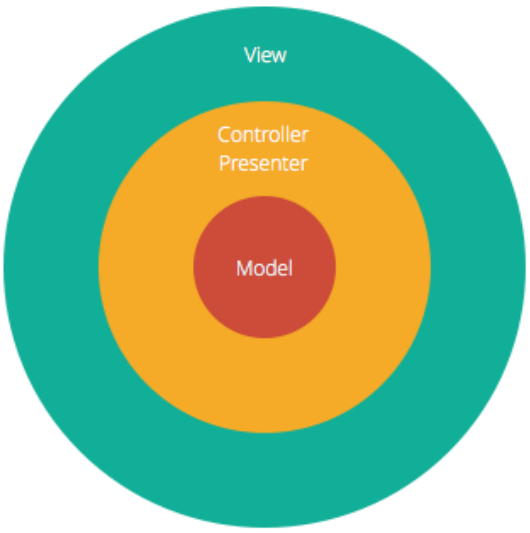
Jedno z kľúčových rozhodnutí je voľba typu architektúry. Tie najpoužívanejšie sú vymenované nižšie podľa článku [5] napísanom na základe knihy[**?**], ktorú napísal Mark



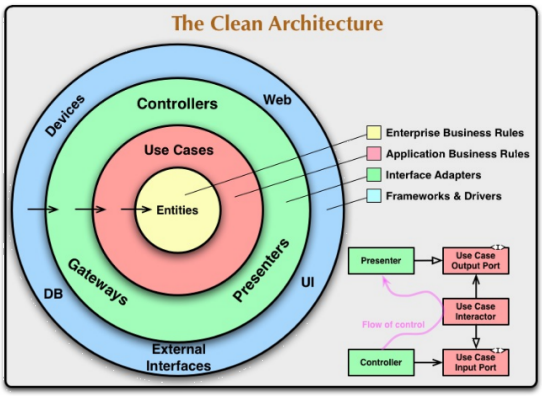
Obrázok 17: Oblasti, potrebné zvažovať pri návrhu[2]

Richards - softvér architekt s 30 ročnou skúsenosťou. V článku sa tiež uvádza, že v jednom systéme môže byť použitých viacero typov, čo je dôležitá informácia pri návrhu.

- **N-vrstvová architektúra.** Je to jeden z najpoužívanejších prístupov, pretože je to postavené okolo databázy a veľa aplikácií v biznise prirodzene potrebujú ukladať informácie v tabuľkách. Veľa z najznámejších frameworkov ako Java EE, Drupal a Express sú postavené tak, aby mali túto štruktúru na mysli, takže aplikácie, nimi vytvorené sú automaticky N-vrstvové. Zdrojový kód je usporiadaný tak, že dáta vstupujú na najvyššej úrovni a prepracujú sa cez každú vrstvu až dosiahnú najnižšiu, čo je zvyčajne databáza. Po ceste má každá vrstva svoju úlohu, ako kontrolovanie konzistencie dát alebo formátovanie dát, tak aby ostali konzistentné. Je bežné, že rôzny programátori pracujú nezávisle na jednotlivých vrstvách. MVC (Model-View-



(a) 3 vrstvová architektúra [6]



(b) 4 vrstvová architektúra[4]

Controller) štruktúra, ktorú poskytuje väčšina obľúbených frameworkov je zjavne N-vrstvová architektúra. Nad databázou je model, ktorý často obsahuje biznis logiku

a informácie o type dát databáze. Na vrchu je zobrazovacia vrstva, ktorá často pozostáva z CSS, JavaScript a HTML. V strede je ovládač, ktorý má viacero pravidiel a funkcií na transformáciu dát zo zobrazovacej vrstvy do modelu.

- **Architektúra riadená udalosťami.** Veľa programov trávi väčšinu času čakaním, kým sa niečo stane. Tento fakt špeciálne platí pre systémy, ktoré priamo spolupracujú s ľuďmi, ale rovnako je to bežné aj v oblasti sietí. Architektúra riadená udalosťami pomáha spravovať uvedené fakty tak, že sa vytvorí centrálna jednotka, ktorá prijíma dáta a potom ich deleguje do samostatných modulov, ktoré dáta spracujú. Toto odovzdanie sa nazýva vygenerovanie udalosti. Udalosť je následne spracovaná kódom tzv. event-handler.

- **„Microkernel“ architektúra.** Veľa aplikácií má základnú sadu operácií, ktoré sú znova a znova použité v iných prípadoch, ktoré závisia od aktuálneho typu dát a typu úlohy. Obľúbený nástroj na vývoj Eclipse, napríklad, najskôr otvorí súbory, pridá im poznámky, upraví ich a potom spustí pomocníka na pozadí. Nástroj je vykonávaním týchto operácií známy a s kódom napísaným v jazyku Java na jedno stlačenie tlačítka sa kód skompiluje a spustí. V tomto prípade, základný program na zobrazovanie a upravovanie súboru sú súčasťou microkernel-u. Java kompilátor je extra časť, ktorá je pridaná na podporu základných čŕt microkernel-u. Ostatní vývojári vyvinuli ďalšie časti, aby bolo možné vyvíjať aj v iných jazykoch s inými kompilátormi. Často krát sa kompilátor ani nepoužíva, ale využívajú len funkcie na úpravu súborov. Špeciálne pridaná funkcionalita sa nazýva plug-in. Často je tento prístup nazývaný aj Plug-in architektúra.

- **„Microservice“ architektúra.** Softvér môže byť ako malý slon. Keď je malý je milý a zábavný, ale keď dospeje, je ťažké ho viesť a bráni sa zmene. Návrh Microservice architektúry pomáha vývojárom predísť tomu, aby sa z ich malých programov stali ťažkopádne, monolitické a neflexibilné programy. Preto na miesto vytvárania jedného veľkého programu je cieľ vytvoriť množstvo rozličných malých programov a vždy keď chce niekto pridať funkcionalitu, tak vždy pridať malý program. Tento prístup je podobný Microkernel a udalosťami riadenému prístupu, ale je používaný zvyčajne, keď rozličné úlohy sú ľahko oddeliteľné. Vo veľa prípadoch, rozličné úlohy môžu vyžadovať rôzny čas na spracovanie a môžu sa líšiť v spôsobe použitia. Napríklad servery spoločnosti Netflix, ktoré poskytujú obsah zákazníkom (filmy a videá) majú oveľa väčšiu záťaž v piatok a sobotu večer, takže musia byť pripravený zvýšiť výpočtové a sieťové kapacity. Servery, ktoré sledujú

vrátenie požičaných DVD, na druhej strane, robia svoju robotu cez týždeň hneď ako pošta doručí príchodzie zásielky. Ak sa to implementuje ako oddelené služby, Netflix cloud ich môže nezávisle škálovať podľa dopytu.

- **„Space-based“ architektúra.** Veľa aplikácií, ktoré sú postavené okolo databázy a fungujú správne pokiaľ databáza stíha spracovávať záťaž. Keď však databáza prestane stíhať zapisovať veľa transakcií, celá aplikácia spadne. „Space-based“ architektúra je navrhnutá tak, aby predišla zlyhaniu pri veľkej záťaži rozložením spracovania a ukladania na viacero serverov. Dáta aj zodpovednosť za možnosť zavolania služby sú rozdistribuované na viacero uzlov. Niektorí architekti používajú širokejší pojem „cloud“ architektúra. Táto architektúra podporuje prípady, kedy je ťažké predikovať nárast požiadaviek, ktorý by databáza nestíhala spracovávať. Uchovávanie informácie v RAM umožňuje vykonávať veľa úloh rýchlejšie a zjednodušuje zdieľanie medzi uzlami. Avšak distribuovaná architektúra robí niektoré analýzy komplikovanejšími. Výpočet, ktorý musí prebehnúť cez všetky dáta, napríklad výpočet priemeru alebo vytváranie štatistickej analýzy musí byť rozdelený na podúlohy cez všetky uzly a keď skončia zoskupenie dát je potrebné.

2.2 Základné typy aplikácií

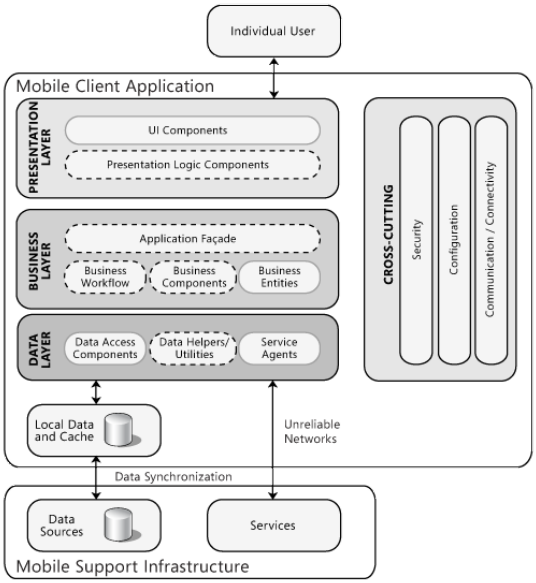
Ďalšie z kľúčových rozhodnutí je voľba typu aplikácie. Na výber podľa článku [3] sú:

- **Mobilná aplikácia 19a.** Aplikácie tohto typu môžu byť vyvíjané ako tenký alebo tučný klient. Mobilná aplikácia vo forme tučného klienta môže podporovať scenáre bez pripojenia alebo s občasným pripojením. Webové aplikácie alebo tenkí klienti podporujú iba scenáre s pripojením. Zdroje, ktorými disponujú mobilné zariadenia sa často ukazujú ako obmedzenie pri návrhu mobilných aplikácií.
  Výhody:
  - Podpora mobilných zariadení.
  - Dostupnosť a jednoduchosť použitia pre používateľov mimo kancelárie.
  - Podpora pre scenáre bez pripojenia a s občasným pripojením.

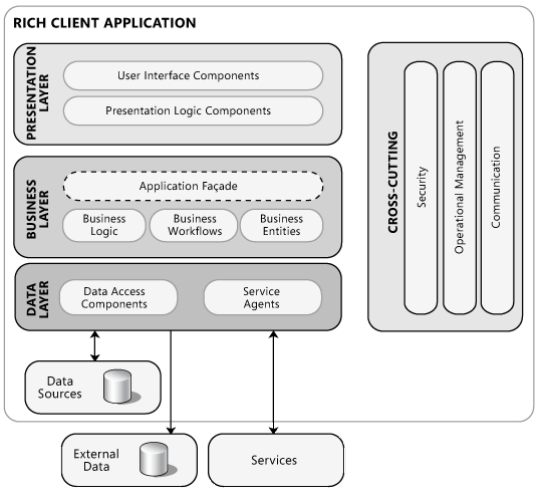
  Na zváženie:
  - Limity ohľadne zadávania údajov a navigácie v aplikácii.
  - Limitovaná šírka zobrazovanej plochy.

- **Tučná klientská aplikácia 19b.** Aplikácie tohto typu sú zvyčajne vyvíjané ako stand-alone aplikácie s grafickým rozhraním, ktoré zobrazuje dáta prostredníctvom rôznych ovládacích prvkov. Tuční klienti sú väčšinou navrhnutí na scenáre bez pripojenia alebo s občasným pripojením, keď potrebujú prístup k vzdialeným dátam alebo funkcionalite.
  Výhody:
  - Možnosť využívať zdroje klienta.
  - Lepšia odozva, bohatá funkčnosť používateľského rozhrania a lepší používateľský zážitok.
  - Dynamická a responzívna interakcia.
  - Podpora scenárov bez pripojenia alebo s dočasným pripojením.

  Na zváženie:
  - Komplikovanejšie nasadenie. Avšak existuje množstvo inštalačných možnosti ako ClickOnce, Windows Installer a XCOPY je dostupných.
  - Problém s nasadzovaním nových verzií v čase.
  - Závislé od platformy.



(a) Mobilná aplikácia [3]



(b) Clientská aplikácia [3]

Obrázok 19

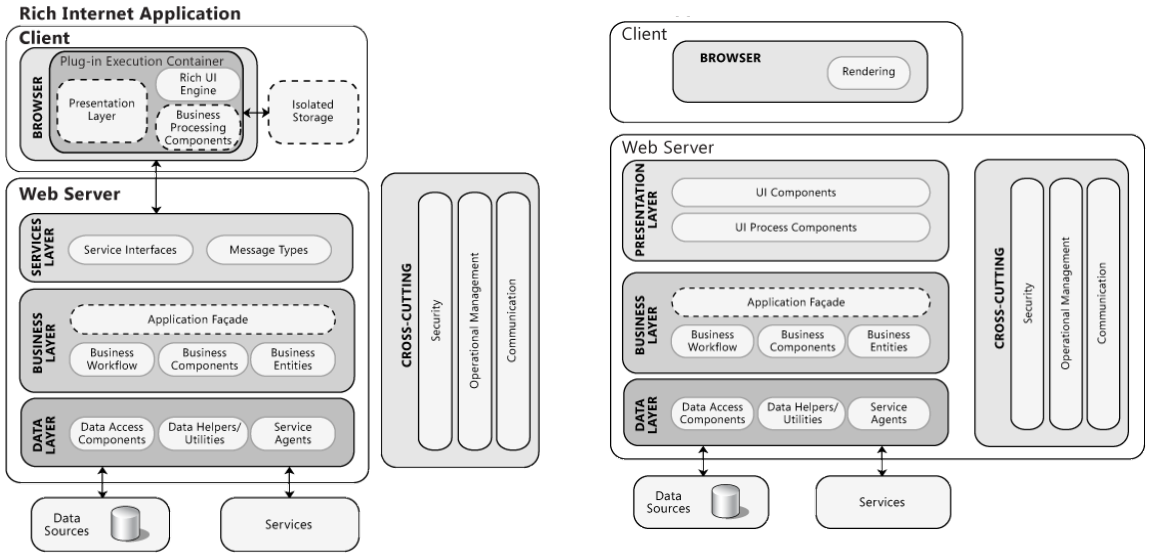
- **Tučná internetová aplikácia 20a.** Aplikácie tohto typu sú vyvíjané tak, aby podporovali viacero platforiem a prehliadačov tak, že zobrazujú multimédia alebo iný grafický obsah. Tučné internetové aplikácie bežia v prehliadači, čím sú obmedzený pristupovať k niektorým zdrojom klienta.
  Výhody:
  - Rovnaké schopnosti užívateľského rozhrania ako tuční klienti.
  - Podpora multimédií a stream medií
  - Jednoduché nasadzovanie s rovnakými možnosťami distribúcie ako weboví klienti.
  - Jednoduchý upgrade na novú verziu.
  - Podpora cez väčšinu platforiem a prehliadačov.

  Na zváženie:
  - Kladie vyššie nároky na klienta ako webové aplikácie.
  - Obmedzenia na využívanie zdrojov klienta oproti tučnej klientskej aplikácii.
  - Vyžaduje mať na klientovi nasadený vhodný runtime framework.

- **Webová aplikácia 20b.** Aplikácie tohoto typu zvyčajne podporujú scenáre s pripojením, podporujú viacero prehliadačov, ktoré bežia na rôznych operačných systémoch.
  Výhody:
  - Široká dostupnosť pre väčšinu platforiem a užívateľské rozhranie je vytvárané prostredníctvom štandardov.
  - Jednoduchosť nasadenia a správy zmien.

  Na zváženie:
  - Závislé od nepretržitého sieťového spojenia.
  - Komplikácie s poskytnutím bohatého (rich) užívateľského rozhrania.

- **Servisná aplikácia 21.** Služby vystavujú zdieľanú biznis funkcionalitu a umožňujú klientom pristupovať k nim z lokálneho alebo vzdialeného systému. Operácie služieb sú volané prostredníctvom správ založených na XML schémach, posielaných cez transportnú vrstvu. Cieľom týchto aplikácií je dosiahnutie voľných väzieb medzi



(a) Tučná internet aplikácia [3]

(b) Web aplikácia [3]

Obrázok 20

klientom a serverom.

Výhody:

- Interakcie medzi klientom a serverom je prostredníctvom voľných väzieb.
- Môže byť použitá rôznymi nezávislými aplikáciami.
- Podpora pre interoperabilitu.

Na zváženie:

- Nie je podpora pomocou užívateľského rozhrania.
- Závisí od sieťového pripojenia.

2.2.1 Servisný aplikácia

TODO: soap rest point to point

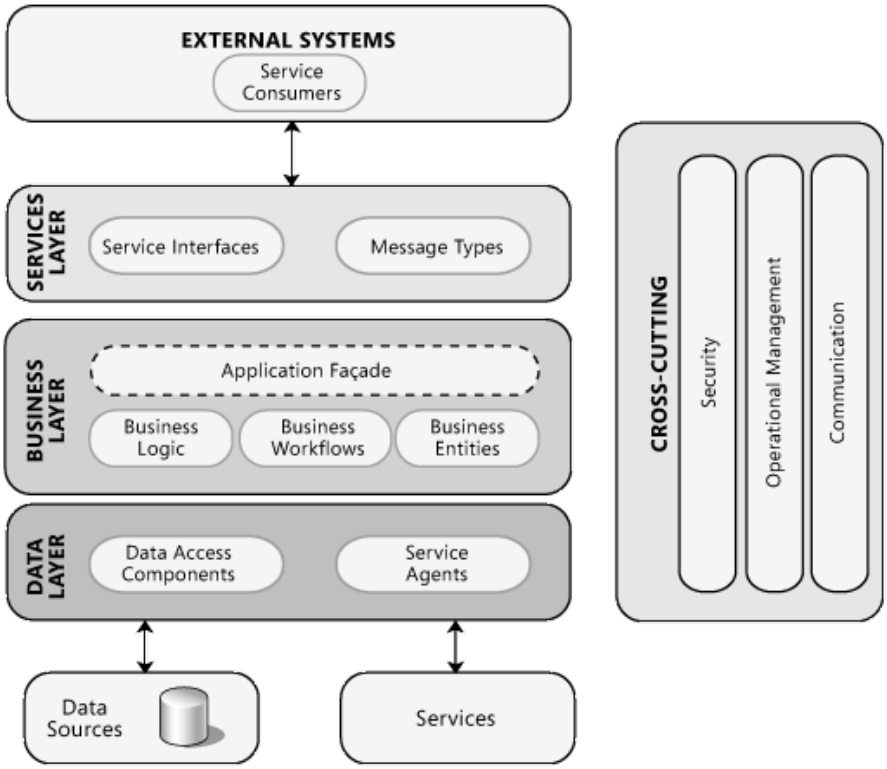
2.3 Internet vecí - IoT

TODO: Definícia

TODO: 3 súčasti senzory akcne cleny obrazok vrstiev gateway cloud

TODO: iot azure a ine riesenia a obrazky architektury

TODO: Preco automatizacia - lebo senzory, zbernice a embedded zariadenia v procesoch



Obrázok 21: Aplikácia poskytujúca službu [3]

TODO: ale ved to uz jerozdiel medzi existujucimi priemyselnymi rieseniami a iot http://electronicdesign.com/iot/iot-industrial-way

2.3.1 Oblasti využitia

TODO: Inteligentná budova TODO: dalsie oblasti = obrazok oblasti + citat google

2.4 Činnosti súvisiace s architektúrou

TODO: architektura aplikacie zahrna tieto cinnosti

2.4.1 Návrh

TODO: nástroje UML, enterprise + pohlady TODO: rozdiely is a iot = aj hw design

2.4.2 Vývoj

TODO: jazyky Java, C#, PHP, JavaScript, Python TODO: Trendy angular a ine technologie na RIA a ine TODO: rozdiely is a iot - pripajanie na senzory a pripajanie na Big Data

2.4.3 Údržba

TODO: ake nastroje nagios a dalsie google TODO: rozdiely starost o hw zariadenia TODO: co je continuous integration a ako zabezpecit

3 Implementácia riešenia

TODO: implementacia sa sklada z viacerych casti

3.1 Popis IoT prostredia

TODO:popis vera zbernica prvky nevyhody vo OneNote

3.1.1 Tvorba modulov

TODO: plugin architektura RSS reader

3.2 CaaS

TODO: vysvetlit + doplnit motivaciu z uvodu

3.2.1 Implementácia

TODO: micro service architektura TODO: service aplikacia

3.2.2 Integrácia

TODO: novy modul

3.2.3 Experiment

TODO: popis zapojenia TODO: identifikacia TODO: grafy TODO: pripad pouzitia udrziavanie intenzity pripad pouzitia teplota - upozornit na ekvitermiku pripad pouzitia

Záver

TODO: zaver

Zoznam použitej literatúry

[1] 10 iot predictions for 2016 - page: 1 | crn. http://www.crn.com/slide-shows/networking/300079629/10-iot-predictions-for-2016.htm. Accessed on 14-06-2016).

[2] Chapter 1: What is software architecture? https://msdn.microsoft.com/en-us/library/ee658098.aspx. (Accessed on 16-06-2016).

[3] Chapter 20: Choosing an application type. https://msdn.microsoft.com/en-us/library/ee658104.aspx. (Accessed on 16-06-2016).

[4] The clean architecture | 8th light. https://blog.8thlight.com/uncle-bob/2012/08/13/the-clean-architecture.html. (Accessed on 17-06-2016).

[5] How to choose the right software architecture: The top 5 patterns. http://techbeacon.com/top-5-software-architecture-patterns-how-make-right-choice. (Accessed on 17-06-2016).

[6] javascript - angularjs: Understanding design pattern - stack overflow. http://stackoverflow.com/questions/20286917/angularjs-understanding-design-pattern/30745329#30745329. (Accessed on 17-06-2016).

[7] Stabilita, kvalita a presnosť regulácie | www.senzorika.leteckafakulta.sk. http://www.senzorika.leteckafakulta.sk/?q=node/298. (Accessed on 15-06-2016).

[8] Balandat, M. Constrained Robust Optimal Trajectory Tracking: Model Predictive Control Approaches. http://hybrid.eecs.berkeley.edu/~max/pubs/pdf/Diploma_Thesis.pdf. Accessed on 12-02-2014).

[9] Bemporad, A., and Morari, M. Robust Model Predictive Control: A Survey, 1999.

[10] Bradley, A., Bennick, A., and Salciccioli, M. Model Predictive Control. https://controls.engin.umich.edu/wiki/index.php/MPC/. Accessed on 12-02-2014).

[11] Řehoř, J. Návrh explicitního prediktivního regulátoru s využitím Multi-Parametric Toolboxu pro Matlab. http://www2.humusoft.cz/www/papers/tcp07/rehor.pdf. Accessed on 12-02-2014).

[12] Grimble, M. J. Robust industrial control systems: Optimal design approach for polynomial systems, 2006.

[13] Kozák, ., and Dúbravský, J. On-line predictive controller for gas turbine, 2013.

[14] Takács, G., and Rohaľ-Ilkiv, B. Model Predictive Vibration Control, 2012.