Teória automatického riadenia III.

G. Takács, G. Batista

Ústav automatizácie, merania a aplikovanej informatiky
Strojnícka fakulta, Slovenská technická univerzita

Ak kvadratický problém definujeme ako,

$$\min_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{g}, \tag{1}$$

$$s.t. \quad \mathbf{A}_c\mathbf{x} \leq \mathbf{b}_0 \tag{2}$$

$$\mathbf{A}_e\mathbf{x} = \mathbf{b}_e \tag{3}$$

$$\tag{4}$$

máme k dispozícií viacero softvérových nástrojov na jeho riešenie.

V softvérovom prostredí MATLAB môžeme použiť funkciu `quadprog` ktorá rieši problém tvaru z rovníc 1 až 3. Funkcia `quadprog` poskytuje možnosť riešiť kvadratický problém metódami

- vnútorného bodu (interior point),
- aktívnej množiny (active set),
- regiónu dôvery

Prispôsobiteľnosť tejto funkcie nám spôsobuje že nieje ideálna na robenie výpočtov kvadratického problému v aplikácií prediktívneho riadenia s veľmi krátkymi periódami vzorkovania.
Funkcia `quadprog` je zložitá m-funckia obsahujúca iné m-funkcie a funckie v binárnom formáte, ktorú nieje možné preložiť ako celok do binárneho kódu. Taktiež nepatrí medzi funkcie kompatibilné so Simulinkom.

Syntax použitia tohoto príkazu je nasledovný:

```
[X,fval,exitflag,output,lambda] =
                  quadprog(H,f,A,b,Aeq,beq,LB,UB,X0,options)
```

```
min 0.5*x'*H*x + f'*x
subject to:  A*x <= b,
             Aeq*x = beq,
             LB <= X <= UB.
```

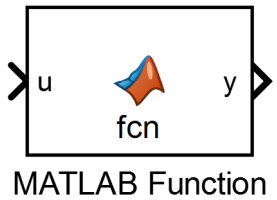
kde argumenty od `H` po `UB` prislúchajú definovaným ohraničeniam problému, `X0` je nami daná počiatočná hodnota na štart algoritmu a objekt `options` obsahuje nastavenia riešiča. Tieto nastavenia sa vytvárajú pomocou príkazu `optimoptions`. Ľubovoľné argumenty je možné vynechať dosadením prázdnej množiny `[]` na miesto vybraného argumentu.

Výstupné hodnoty funkcie sú

- `x`, vektor optimálnych vstupov
- `fval`, funkčná hodnota kriteriálnej funkcie
- `exitflag`, informácia o skončení algoritmu
- `output`, informácia o optimalizácií
- `lambda`, Lagrangeove súčinitele výsledku

Ako použiť `quadprog` v Simulinku?

Simulink pracuje na princípe interpretovania blokovej schémy a následnej kompilácie do binárneho kódu. Keďže `quadprog` nieje dostupný ako blok v knižnici Simulinku, tak ho tam je potrebné vložiť cez blok MATLAB Function Block.



Blok MATLAB Function v Simulinku

Toto nám umožnuje spúšťať písaný MATLAB kód priamo v Simulinku.

Tento kód však musí byť kompilovateľný Simulinkom. Preto je v ňom treba správne alokovať pamäť pre jeho vnútorné a výstupné premenné a taktiež treba dbať na použitie kompatibilných MATLAB funkcií. V prípade, keď nieje možné použiť takúto kompatibilnú funkciu, existuje možnosť ako povedať Simulinku aby sa niektore funkcie nekompilovali a namiesto toho sa zavolal MATLAB interpreter. Toto sa robí pomocou príkazu `coder.extrinsic`.

Príklad takéhoto kódu je nasledovný:

```
function u = fcn(H,G,x,Ac,b0)
%#codegen

u=double(ones(30,1));

% MPC with constrains
coder.extrinsic('optimset','quadprog');
options = optimset('Display','off','Algorithm','active-set');
[u_ast,f]=quadprog(H,G*x,Ac,b0,[],[],[],[],[],options);

u=u_ast;
```

Ďalšia možnosť ako riešiť kvadratický problém je použitie softvérového balíku qpOASES (quadratic programming Online Active-Set Strategy)[Ferreau et al.2014]. Tento softvér rieši problém v tvare:

$$\min_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{g}(w_0), \tag{5}$$

$$s.t. \quad \mathbf{lbA}(w_0) \leq \mathbf{A}\mathbf{x} \leq \mathbf{ubA}(w_0), \tag{6}$$

$$\mathbf{lb}(w_0) \leq \mathbf{x} \leq \mathbf{ub}(w_0) \tag{7}$$

kde je Hessian symetrická a pozitívne (semi-)definitná matica a vektor gradientu spolu s vektormi ohranicenia sú lineárne závislé na parametri w_0

Softvérový balík qpOASES je písaný v jazyku C++, čo ho umožnuje použiť na širšej škále platforiem. Vyžaduje si však menšie úprvy nato aby bol použiteľný. Tento riešič je používa stratégiu aktívnej množiny a je optimalizovaný pre podávanie vysokých výkonov pomocou tejto metódy.
Jeho najväčšou výhodou je použitie sekvenčného výpočtu kvadratického problému, tzv. hotstart. To znamená že ak je riešenie nasledujúceho problému v sekvencií lineárne závislé na predošlom výsledku, tak tento výsledok môžeme použiť ako počiatočný odhad nasledujúceho problému. Z tohoto je jasné že táto metóda je vhodná pre použitie v prediktívnom riadení, keďže to umožňuje podstatne znížiť počet iterácií potrebných k získaniu výsledku a tým skrátiť výpočtový čas.

qpOASES ponúka tri triedy použitia tohoto riešiča:

- QProblem - zjednodušený štandardný tvar
- QProblemB - jednoduché ohraničenia
- SQProblem - štandardný tvar

Trieda QProblem rieši problém v štandardnom tvare, v prípade že sa matice H a A nemenia.

$$\begin{aligned} \min_{\mathbf{x}} \quad & \tfrac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{g}(w_0), \\ s.t. \quad & \mathbf{lbA}(w_0) \leq \mathbf{A}\mathbf{x} \leq \mathbf{ubA}(w_0), \\ & \mathbf{lb}(w_0) \leq \mathbf{x} \leq \mathbf{ub}(w_0) \end{aligned}$$

Trieda QProblemB rieši kvadratický problém v ktorom existujú iba jednoduché ohraničenia a Hessian je nemenný

$$\min_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{g}(w_0),$$
$$\mathbf{lb}(w_0) \leq \mathbf{x} \leq \mathbf{ub}(w_0)$$

Trieda SQProblem rieši štandardný problém v ktorom je možnosť zmeny všetkých parametrov z kroku na krok. Táto varianta je vhodná pre použite napríklad v regulácií časovo variantných systémov s meniacimi sa ohraničeniami.

$$\begin{aligned} \min_{\mathbf{x}} \quad & \tfrac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{g}(w_0), \\ s.t. \quad & \mathbf{lbA}(w_0) \leq \mathbf{A}\mathbf{x} \leq \mathbf{ubA}(w_0), \\ & \mathbf{lb}(w_0) \leq \mathbf{x} \leq \mathbf{ub}(w_0) \end{aligned}$$

qpOASES ponúka rôzne rozhrania pre použitie tohoto riešiča. Na výber máme z:

- C
- MATLAB
- Simulink
- Octave
- Scilab
- python

Pre použitie v prostredí MATLAB/Simulink je potrebné mať nainštalovaný a nakonfigurovaný compiler C++. qpOASES je dostupný zo stránky `https://projects.coin-or.org/qpOASES/` zadarmo.

Balíček funkcií pre MATLAB obsahuje funkcie:

- `qpOASES(...);` pre riešenie štandardného kvadratického problému
- `qpOASES_sequence(...);` pre riešenie sekvenčného kvadratického problému
- `qpOASES_options(...);` pre vytvorenie nastavení riešiča

Syntax štandardného kvadratického problému qpOASES je nasledovný:

```
min    1/2*x'Hx + x'g
s.t.   lb   <=   x <= ub
       lbA <= Ax <= ubA   {optional}
```

```
[x,fval,exitflag,iter,lambda,auxOutput] =
   qpOASES( H,g,A,lb,ub,lbA,ubA{,options{,auxInput}} )
```

kde argumenty od `H` po `ubA` prislúchajú kvadratickému problému a jeho ohraničeniam. Dodatočné argumenty sú nastavenia riešiča a vedlajšie vstupy ktoré sa postúpia riešiču. Matica `H` musí byť symetrická a vektory od `g` po `ubA` musia byť stĺpcové.
Maticu `A` stačí inicializovať pre jednu stranu ohraničenia, napríklad pre horné ohraničenie, keďže qpOASES sám vytvorí potrebnú maticu pre celkové ohraničenie z hornej aj dolnej strany. Je taktiež možné inicializovať zjednodušený problém ktorý neobsahuje zovšeobecnené hraničné podmienky vynechaním argumentov `A, lbA, ubA`.

Výstupné hodnoty funckie qpOASES sú:

- `x`, optimálne primárne riešenie (ak `exitflag==0`)
- `fval`, optimálna hodnota cieľovej funkcie (ak `exitflag==0`)
- `exitflag`, informácia o spôsobe ukončenia algoritmu
- `iter`, počet iterácií ktoré si vyžadovalo riešenie]
- `lambda`, Lagrangeove súčinitele výsledku
- `auxOutput`, štruktúra obsahujúca dodatočné výstupy

Nastavenia riešiča je možné vytvoriť príkazom `qpOASES_options(...)` v ktorom je možné nastaviť ľubovoľné parametre. qpOASES ale ponúka tri základné skupiny nastavení ktoré môžeme využiť. Sú to:

- `options = qpOASES_options( 'default');` pre štandardné nastavenie
- `options = qpOASES_options( 'reliable');` pre nastavenie spoľahlivej konfigurácie riešiča
- `options = qpOASES_options( 'MPC');` pre nastavenie rýchleho riešenia kvadratického problému

Sekvenčný kvadratický problém je možné vypočítať pomocou príkazu `qpOASES_seuence(...)`. Priebeh jeho riešenia je možné rozdeliť do troch skupín.

- incializácia problému (vytvorenie objektu kvadratického programovania)
- sekvenčné riešenie (nadväzujúce problémy sa riešia pomocou výstupu predošlého)
- ukončenie riešenia kvadratického problému (uvoľnenie operačnej pamäte)

Sekvenčný problém inicializujeme podobne ako štandardný problém. Avšak je potrebné pridať ako výstupný argument do ktorého uložíme objekt kvadratického problému. Syntax je nasledovný:

```
[QP,x,fval,exitflag,iter,lambda,auxOutput] = ...
 qpOASES_sequence('i',H,g,A,lb,ub,lbA,ubA{,options{,auxInput}})
```

kde parameter `QP` je spomenutý objekt a argument `'i'` oznamuje funkcií že sa jedná o inicializáciu problému.

V ďalšom kroku máme na výber z troch možností. V závislosti od prvého argumentu môžeme spustiť:

- `qpOASES_sequence('h',QP,g,lb,ub,lbA,ubA,options)`, štandardný sekvenčný výpočet, tzv. hotstart, v ktorom aktualizujeme gradient a ohraničenia problému
- `qpOASES_sequence('m',QP,H,g,A,lb,ub,lbA,ubA,options)`, sekvenčný výpočet s premenlivými maticami `H,g` a taktiež s aktualizovanými ohraničeniami
- `qpOASES_sequence('e',QP,g,lb,ub,lbA,ubA,options)`, sekvenčný problém s rovnostnými ohraničeniami

Ďalej je potrebé pridať ako argument funkcie objekt kvadratického problému `QP`. Charakteristikou tohoto sekvenčného výpočtu je to že počiatočný odhad výsledku je vlastne výsledok predošlého kroku.

V poslednom kroku je nutné zavolať funkciu vo forme

```
qpOASES_sequence( 'c',QP )
```

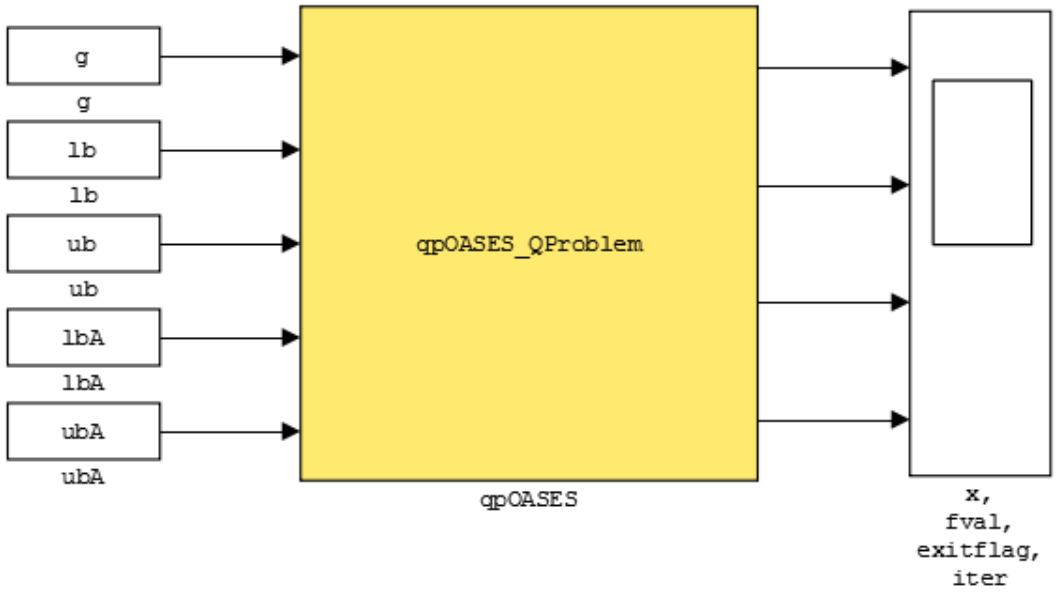
nato aby sa zmazal objekt `QP` a uvolnila sa alokovaná pamäť.

Pre použitie v prostredí Simulink je najskôr treba nastaviť simulačné parametre riešiča modelu. Je nutné nastaviť diskrétny riešič s fixným integračným krokom.
Ďalej treba tiež skompilovať S-funkciu qpOASES do binárneho formátu. Takýmto spôsobom dostaneme tri funkcie:

- `QProblem`
- `QProblemB`
- `SQProblem`

Pri týchto však nemáme k dispozícii sekvenčný typ riešenia.

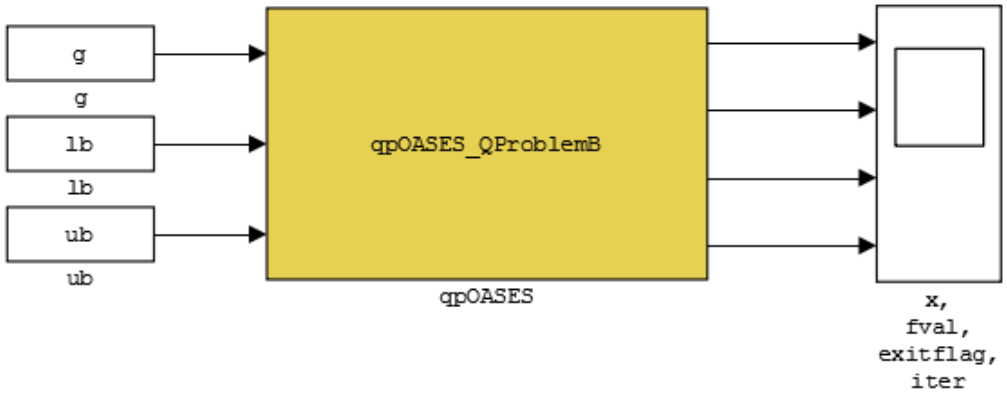
Štandardná forma qpOASES má podobu:



Blok QProblem v Simulinku

do ktorej vstupujú vektory ohraničenia. Matice `H`, `A` sú vnútorné parametre funkcie ktoré treba explicitne definovať.

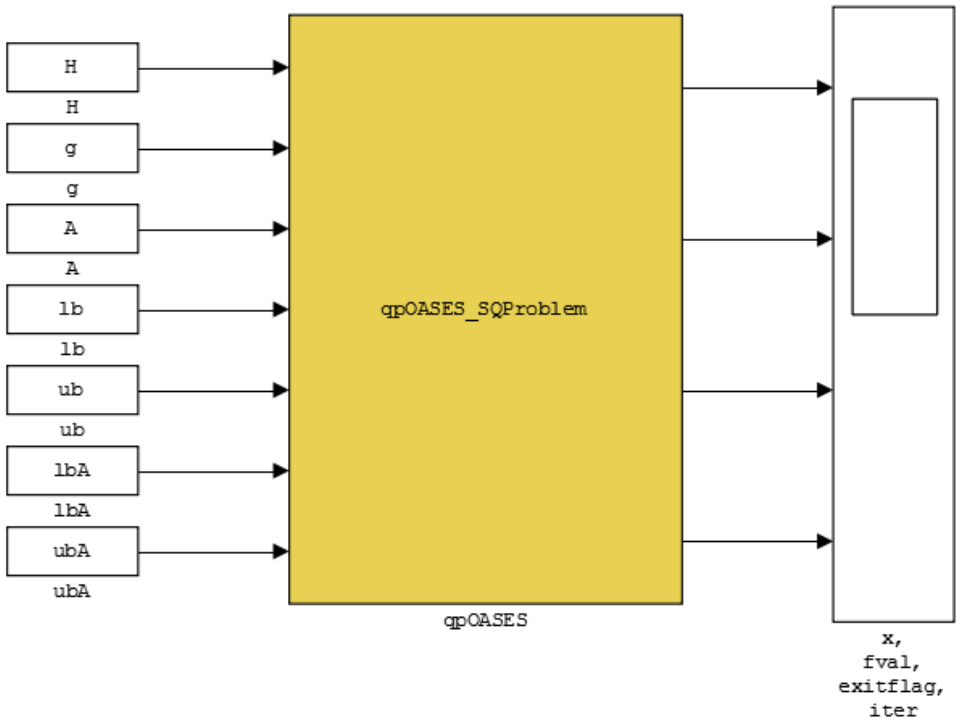
Forma qpOASES s jendoduchými ohraničeniami má podobu:



Blok QProblemB v Simulinku

do ktorej vstupujú vektory ohraničenia. Matica `H` je vnútorný parameter funkcie ktorú treba explicitne definovať.

Posledná forma bloku má podobu:



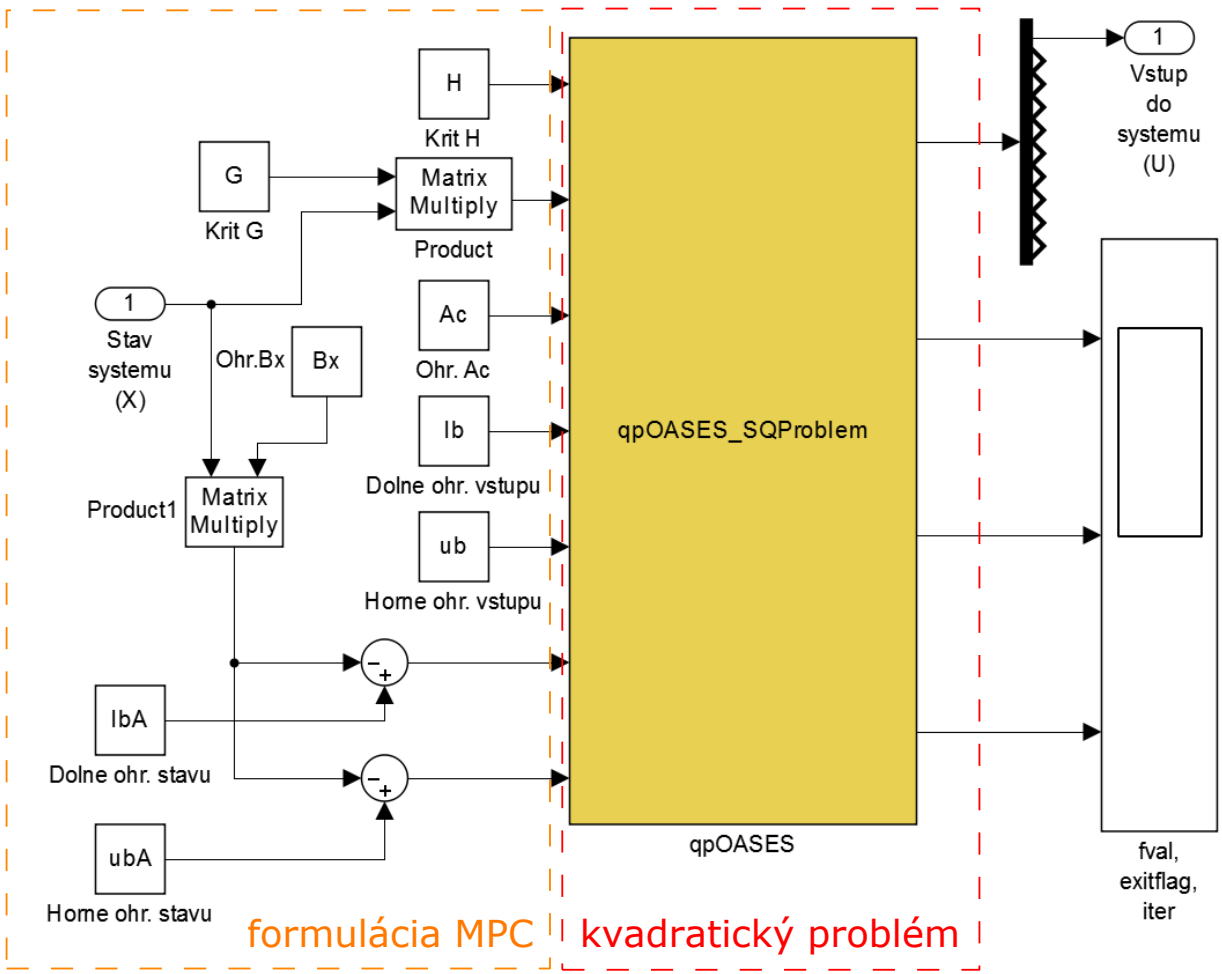
Blok SQProblem v Simulinku

V tomto bloku niesú definované žiadne vnútorné parametre. Všetky nám prichádzajú do bloku ako premenlivé argumenty

Je potrebné poznamenať že v tomto prípade je treba všetky vstupné parametre premeniť do vektorovej podoby, t.j. matice `H, A` je nutné prepísať do jedného vektora.
Ďalej treba spomenúť že niektoré parametre tohoto bloku je potreba nastaviť ešte pri kompilácií S-funkcií (QProblem.cpp). Tieto parametre sú:

- vzorkovacia perióda (-1 pre dedenú zo Simulinku)
- počet vstupov systému, t.j. výstupov bloku QProblem
- maximálny počet iterácií
- typ používaného Hessianu

Na nasledujúcom diagrame je zobrazený príklad použitia qpOASES pre všeobecný problém MPC.



Príklad MPC v Simulinku

Bibliografia I

Ferreau, H., Kirches, C., Potschka, A., Bock, H., and Diehl, M. (2014).
qpOASES: A parametric active-set algorithm for quadratic programming.
Mathematical Programming Computation, 6(4):327–363.