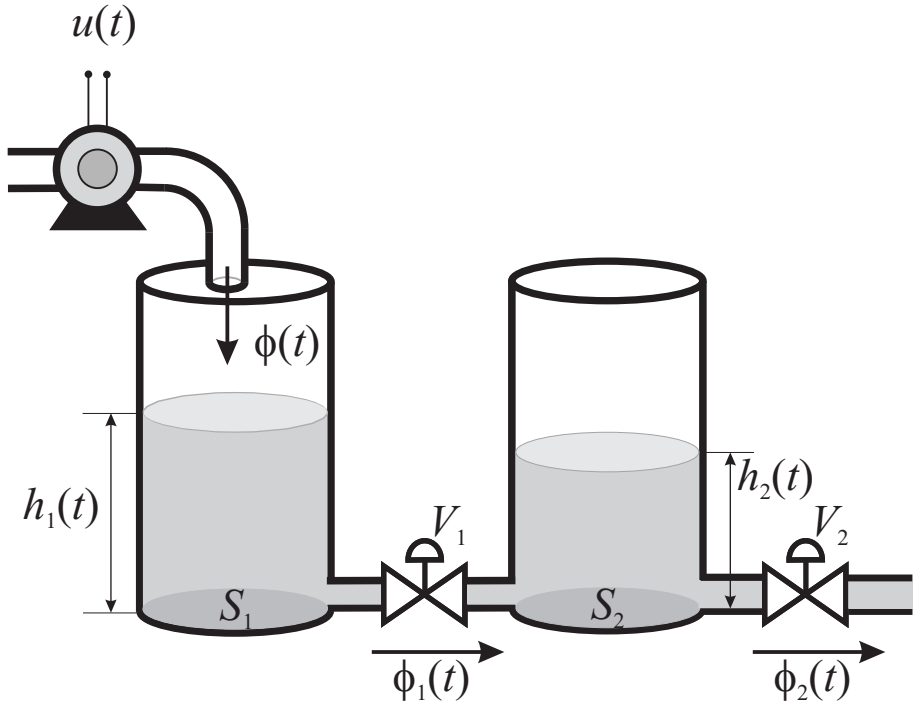
Laboratorijske vaje
Računalniška simulacija
2012/13

3. laboratorijska vaja

Poglobljena uporaba orodja Simulink

Pri tej laboratorijski vaji boste spoznali nekatere napredne funkcije orodja Simulink, kot so iskanje ravnovesnih točk, linearizacija sistemov in optimizacija. Znova bomo obravnavali sistem dveh shranjevalnikov, ki je prikazan na sliki 1.



Slika 1: Shematski prikaz obravnavanega sistema dveh shranjevalnikov

Ventila imata korenski karakteristiki. Številčne vrednosti konstant so:

- $\rho = 1000$ kg/m^3,
- $S_1 = S_2 = 0{,}01$ m^2,
- $k_{V1} = 0{,}1$ kgm^{-1}s$^{-1/2}$ in
- $k_{V2} = 0{,}05$ kgm^{-1}s$^{-1/2}$.

Iskanje ravnovesnega stanja

Za iskanje ravnovesnih stanj sistema se uporablja funkcija `trim`. Ravnovesna stanja so tista stanja, kjer sistem »ostane za vedno«. V primeru časovno zveznih sistemov so ravnovesna stanja točke v prostoru stanj, kjer je odvod vektorja stanj enak 0 (zato se stanje nikamor ne premakne). Kadar imamo izolirana ravnovesna stanja (to so ravnovesna stanja, v okolici katerih ni drugih ravnovesnih stanj), jih lahko s funkcijo `trim` poiščemo. Če je takšnih izoliranih stanj več, potem primerno izberemo začetne pogoje za vhode in stanja in funkcija najde ravnovesna stanja v bližini. Če pa ravnovesna stanja niso izolirana, potem lahko funkciji `trim` predpišemo natančno vrednost vhodov, izhodov in/ali stanj. To storimo tako, da funkcijo `trim` kličemo na naslednji način:

```
[X,U,Y,DX]=trim('SYS',X0,U0,Y0,IX,IU,IY)
```

Če na primer pri zgornjem klicu `IX` nastavimo na `[3,4]`, želimo, da funkcija poišče tista ravnovesna stanja, ki imajo tretji in četrti element vektorja stanj natančno enak tretjemu in četrtemu elementu vektorja `X0`. Na podoben način lahko z nastavitvijo argumenta `IX` na `[]` dosežemo, da `X0` sicer predpisuje začetne vrednosti vektorja stanj pri iskanju ravnovesnih stanj, hkrati pa nobenega elementa ravnovesnih stanj ne predpišemo. Povsem enako vlogo pri predpisovanju začetnih in ravnovesnih vrednosti vhoda in izhoda imajo argumenti `U0`, `IU`, `Y0` in `IY` (ker ima sistem v splošnem lahko več vhodov in/ali izhodov, so vsi ti argumenti v splošnem vektorji).

Linearizacija sistema

Linearizacija je postopek, s katerim poiščemo linearni sistem, ki se v okolici določene točke obnaša podobno kot obravnavani (nelinearni) sistem. Potreben pogoj za izvedbo linearizacije v okolici določene točke je, da obravnavamo ravnovesno točko oz. stanje. V našem primeru smemo torej izvesti linearizacijo sistema okrog katerekoli točke, ki leži na statični karakteristiki sistema. Za linearizacijo sistema, ki ga opisuje shema v orodju Simulink, se uporablja funkcija `linmod`. Funkciji moramo povedati, kaj mi smatramo kot vhod oz. izhod iz sistema. To storimo z uporabo blokov `In1` in `Out1` v simulacijski shemi. Linearizacijo izvedemo z naslednjim funkcijskim klicem:

```
[A,B,C,D]=LINMOD('ime_sheme',X,U)
```

pri čemer matrike `A`, `B`, `C` in `D` predstavljajo matrike zapisa linearnega časovno nespremenljivega sistema v prostoru stanj.

Optimizacija

Optimizacija je postopek, pri katerem poiščemo optimalne vrednosti parametrov. Ključno je seveda, katere parametre spreminjamo in kakšna je mera optimalnosti. Poleg tega je pomembno, kako poteka postopek iskanja, in tudi, če je prostor iskanja parametrov kakorkoli omejen. Najbolj enostavna funkcija, ki izvaja optimizacijo je `fminsearch`. Ta funkcija izvaja optimizacijo brez omejitev po metodi Nelder-Mead. Funkcijo kličemo na naslednji način:

```
[par_fin,kf_fin] = fminsearch('krit_fun',par_init)
```

pri čemer je `krit_fun` m-funkcija, ki izračunava vrednost kriterijske funkcije ob vsakokratni vrednosti parametrov, `par_init` in `par_fin` sta vektorja začetne in končne vrednosti parametrov, `kf_fin` pa je končna (minimalna) vrednost kriterijske funkcije. Kriterijska funkcija ima naslednji vmesnik:

```
function vrednost_kf = krit_fun(par)
```

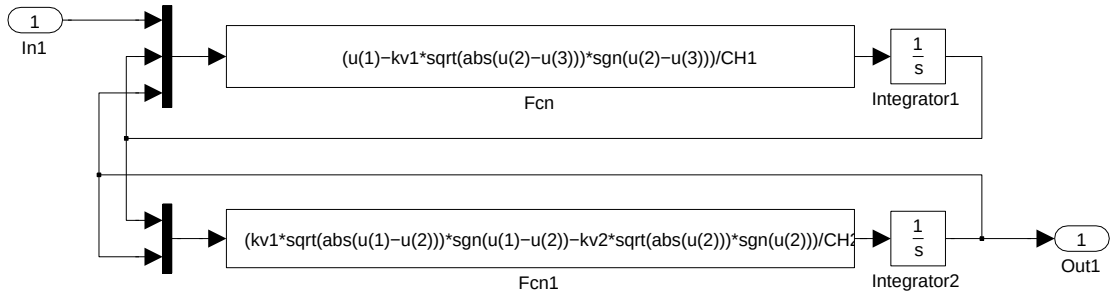
Pri zgornjem klicu je `par` trenutna vrednost vektorja parametrov, `vrednost_kf` pa je pripadajoča vrednost kriterijske funkcije, ki jo mora izračunati in vrniti funkcija.

V našem primeru bomo iskali vrednosti sistemskih parametrov, pri katerih sta dva sistema čimbolj podobna. Seveda je potrebno definirati mero podobnosti – v našem primeru bo to največja vrednost absolutne razlike med odzivoma obeh sistemov na enak vhodni signal.

Naloge

1. Najprej narišite statično karakteristiko sistema, katerega vhod je $\phi(t)$, izhod pa h_2. Pripadajoča simulacijska shema, ki smo jo realizirali pri prvi laboratorijski vaji, je prikazana na sliki 2. V zanki spreminjajte vrednost vhodnega pretoka od 0 do 0,05 kg/s v korakih po 0,001 kg/s. Pri vsakokratnem prehodu skozi zanko kličite funkcijo `trim`, kateri je potrebno fiksirati vhodni signal na vrednost, ki jo spreminjamo v zanki. Funkcija vrne pripadajočo vrednost stanj in izhoda v tej ravnovesni točki. Funkcija naj izriše statično karakteristiko, ki na abscisni osi prikazuje vrednost vzbujalne veličine, na ordinatni osi pa vrednost odziva v ustaljenem stanju.

2. Linearizirajte dani sistem v okolici dveh delovnih točk, in sicer 0,01 kg/s in 0,02 kg/s. Iz dobljenih matrik zapisa sistema v prostoru stanj sestavite dva objekta v Matlabu (funkcija `ss`), ki ju kasneje pretvorite še v zapis s prenosno funkcijo (funkcija `tf`). Na istem grafu prikažite odziva obeh sistemov na

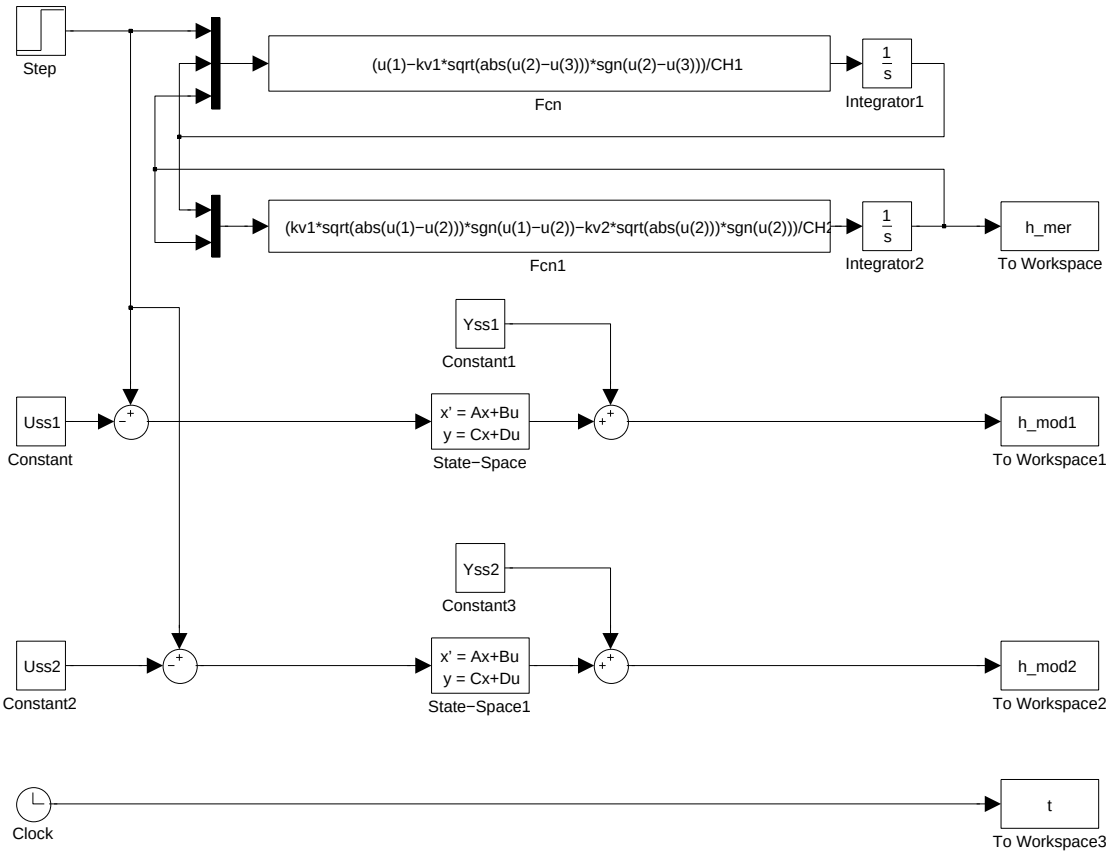


Slika 2: Simulacijska shema pri 1. nalogi

enotino stopnico (funkcija `step`). Poiščite pole (funkcija `pole`), ničle (funkcija `zero`) in enosmerno ojačenje (funkcija `dcgain`) obeh sistemov. Poskusite ugotoviti relacije med izračunanimi lastnostmi obeh sistemov.

3. Pri tej nalogi boste primerjali odzive vseh treh sistemov – originalnega (nelinearnega), lineariziranega pri delovni točki $\phi = 0{,}01$ kg/s in lineariziranega pri delovni točki $\phi = 0{,}02$ kg/s. Ker linearizirani sistem obravnava le odstopanja od delovne točke, je potrebno poskrbeti za ustrezne vrednosti premikov (`Uss` in `Yss`), kot to prikazuje slika 3. Vzbujalni signal naj bo stopničast, pri čemer naj stopnica nastopi ob času 4000 s, celotna simulacija pa naj traja 8000 s. Izvedite dva eksperimenta, pri čemer naj se pri prvem stopničasti vhodni signal začne pri 0,01 kg/s in konča pri 0,011 kg/s, medtem ko naj se pri drugem stopničasti vhodni signal začne pri 0,02 kg/s in konča pri 0,022 kg/s. Na prvem grafu prikažite poteke odzivov vseh treh sistemov pri prvem eksperimentu, na drugem pa odzive sistemov pri drugem eksperimentu.

4. Pri tej nalogi boste izvedli optimizacijo sistema. Primerjali boste odzive dveh sistemov – originalnega (nelinearnega) in takšnega, pri katerem smo korenska ventila nadomestili z linearnima ventiloma s hidravličnima upornostma R_{H1} in R_{H2}. Ker smo s tem dobili linearni sistem, je zopet potrebno zagotoviti ustrezno delovanje sistema v delovni točki, kar dosežemo s premikom signalov za vrednosti `Uss` in `Yss`. Shema celotnega sistema je prikazana na sliki 4. Vzbujalni signal naj bo spet stopničast, pri čemer naj stopnica nastopi ob času 4000 s, vhodni signal naj ima pred nastopom stopnice vrednost 0,01 kg/s, po njej pa 0,011 kg/s. Celotna simulacija naj traja 8000 s. Cilj optimizacije je najti takšni vrednosti R_{H1} in R_{H2}, da bo absolutna vrednost razlike med odzivoma obeh sistemov najmanjša. Da bi zanemarili vpliv napačnega začetnega stanja, je potrebno narediti eno od dvojega – ali ustrezno nastaviti vrednost začetnega stanja ali pa razliko med odzivoma računati šele po nastopu stopnice, ko prehodni pojav na napačno začetno stanje že mine. Še nasvet: Pri tej nalogi bo funkcija `fminsearch` velikokrat klicala kriterijsko funkcijo, ta pa našo simulacijsko shemo, zato svetujem, da z ukazom `warning off all` preprečite izpisovanje opozoril v ukaznem oknu.

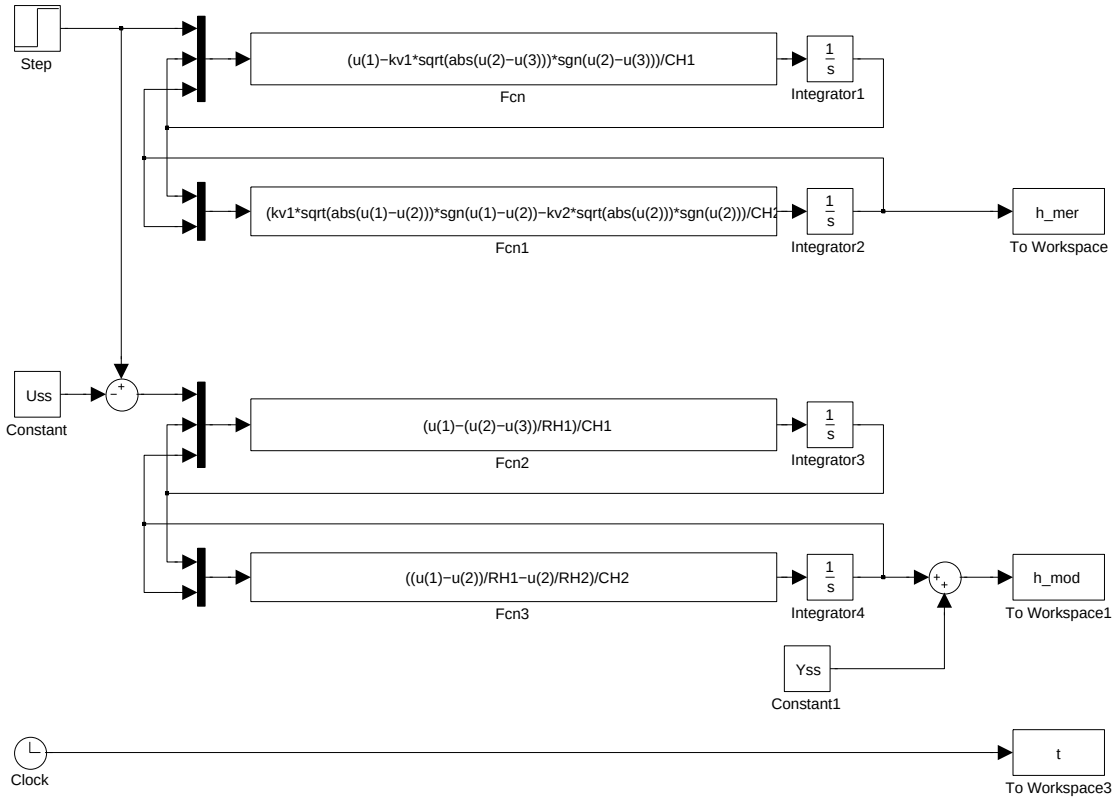


Slika 3: Simulacijska shema pri 3. nalogi

Še kratka pomoč s pisanjem funkcije `krit_fun`. Funkcijsko telo naj se začne z naslednjimi vrsticami:

```
RH1 = par(1);
RH2 = par(2);
if (RH1 <= 0)
    vrednost_kf = inf;
    return;
end
if (RH2 <= 0)
    vrednost_kf = inf;
    return;
end
opts=simset('SrcWorkspace','current','DstWorkspace','current');
```

Zgornje vrstice najprej nastavijo vrednosti spremenljivk v simulacijski shemi, ki ju optimiramo. Potem sledi zelo pomemben kos kode, ki preveri, če je vrednost katere od upornosti negativna ali enaka nič (v takšnem primeru je sistem



Slika 4: Simulacijska shema pri 4. nalogi

nestabilen in izhod sistema bi kmalu dosegel neskončno vrednost, potem pa bi Simulink prekinil izvajanje programa z napako). Zato v tem primeru priredimo kriterijski funkciji vrednost neskončno in se vrnemo v funkcijo `fminsearch`, ne da bi izvedli simulacijo sistema. Nazadnje definiramo še izvorni in ponorni delovni prostor simulacije. Na privzeti način Simulink vrednosti parametrov črpa iz glavnega delovnega prostora, kamor shranjuje tudi vrednosti rezultatov. Zgornja koda pa ponastavi oba delovna prostora na (lokalni) delovni prostor funkcije (`krit_fun`). To pomeni, da moramo po nastavitvi procesnih parametrov in parametrov simulacije klicati simulacijo z ukazom:

```
sim('ime_sheme',[0,tfin],opts)
```

Po izvedbi simulacije mora funkcija `krit_fun` izračunati le še vrednost cenilke – torej največjo vrednost pogreška.