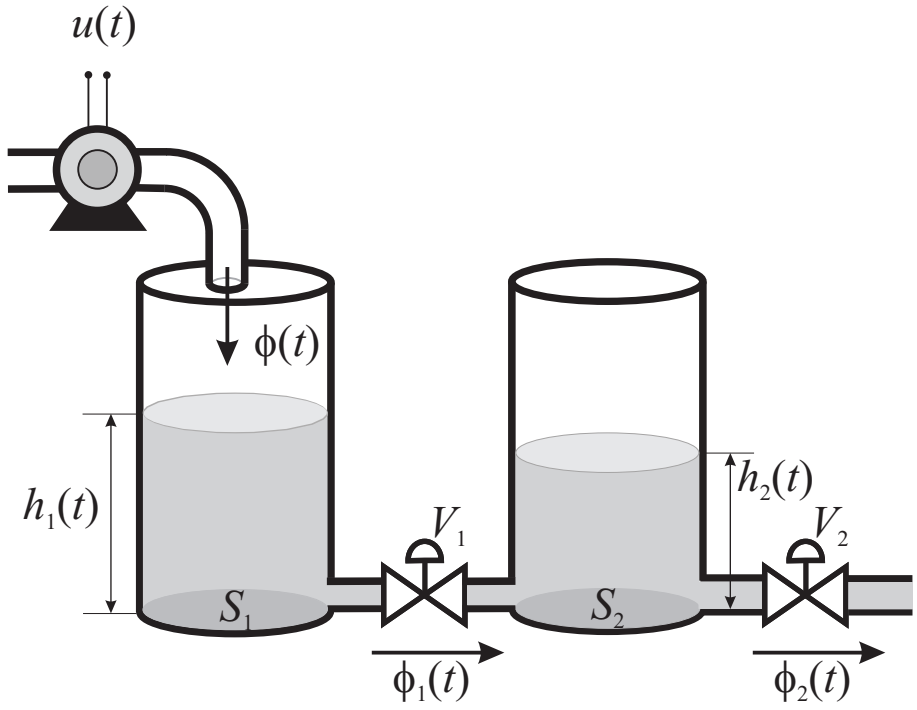
Laboratorijske vaje
Računalniška simulacija
2012/13

2. laboratorijska vaja

Simulacija dinamičnih sistemov s splošnonamenskimi programskimi jeziki

Pri tej laboratorijski vaji boste simulacijo izvedli brez uporabe namenskih knjižnic za simulacijo. Zaradi lažje realizacije boste program napisali v skriptnem jeziku znotraj programskega paketa MATLAB. Podroben opis obravnavane problematike se nahaja v učbeniku v poglavju 5.7 (Simulacija s splošnonamenskimi programskimi jeziki – predvsem razdelka 5.7.1 in 5.7.2). 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,

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

Princip numeričnega reševanja diferencialnih enačb

Simulacijska shema dinamičnega sistema je sestavljena iz več vrst blokov. Osnovni gradniki simulacijskih shem zveznih dinamičnih sistemov so integratorji, seštevalniki in množilniki s konstanto. Če gre za nelinearne sisteme, pa so prisotni tudi množilniki, delilniki in bloki, ki izvajajo nelinearne preslikave. Večino teh blokov zlahka realiziramo v splošnonamenskih programskih jezikih, največjo težavo pa predstavlja izvedba integratorja. Numerična integracija je (razen v izjemno redkih primerih) vedno približna, seveda pa uporaba bolj kompleksnih metod omogoča manjšo napako zaradi numerične integracije. Če zapišemo obravnavani sistem v obliki diferencialne enačbe

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}) \quad \text{z začetnim pogojem } \mathbf{x}(t_0) = \mathbf{x}_0 \tag{1}$$

lahko $\mathbf{x}(t)$ izračunamo tako, da integriramo enačbo (1) po času:

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_{t_0}^{t} \mathbf{f}(t, \mathbf{x}) dt \tag{2}$$

Postopek simulacije izvedemo tako, da celotni časovni interval $[t_0, t]$ razdelimo na množico manjših intervalov. Na vsakem intervalu velja enakost:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{x}) dt \tag{3}$$

Vrednost integrala izračunamo numerično, kar pomeni, da uporabimo kakšno izmed množice integracijskih metod, ki so na voljo. Mi se bomo omejili na enokoračne metode s konstantnim korakom integracije h, kjer velja

$$t_k = t_0 + kh \quad \text{in} \quad t_{k+1} = t_k + h \tag{4}$$

Če razvijemo $\mathbf{x}(t_{k+1})$ v Taylorjevo vrsto in uporabimo le dva člena, dobimo

$$\mathbf{x}(t_{k+1}) \approx \mathbf{x}(t_k) + h\dot{\mathbf{x}}(t_k) = \mathbf{x}(t_k) + h\mathbf{f}(t_k, \mathbf{x}(t_k)) \tag{5}$$

Zlahka lahko pokažemo, da gre za metodo prvega reda (velikost lokalnega pogreška zaradi končnega koraka integracijske metode je sorazmerna s h^2). V nadaljevanju bomo spoznali še dve izboljšani metodi, in sicer Heunovo, ki je drugega reda, in metodo Runge-Kutta, ki je četrtega reda. Pri Heunovi metodi novo vrednost vektorja $\mathbf{x}(t)$ izračunamo po enačbi:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \tfrac{1}{2}\mathbf{k}_1 + \tfrac{1}{2}\mathbf{k}_2 \tag{6}$$

pri čemer pomožni spremenljivki $\mathbf{k}_1$ in $\mathbf{k}_2$ izračunamo z ovrednotenjem odvodne funkcije $\mathbf{f}$:

$$\begin{aligned}\mathbf{k}_1 &= h\mathbf{f}(t_k, \mathbf{x}(t_k)) \\ \mathbf{k}_2 &= h\mathbf{f}(t_k + h, \mathbf{x}(t_k) + \mathbf{k}_1)\end{aligned} \tag{7}$$

Pri metodi Runge-Kutta pa enokoračno napoved vektorja stanj izvedemo na naslednji način:

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \tfrac{1}{6}\mathbf{k}_1 + \tfrac{1}{3}\mathbf{k}_2 + \tfrac{1}{3}\mathbf{k}_3 + \tfrac{1}{6}\mathbf{k}_4 \tag{8}$$

kjer pomožne spremenljivke izračunamo po formulah:

$$\begin{aligned}\mathbf{k}_1 &= h\mathbf{f}(t_k, \mathbf{x}(t_k)) \\ \mathbf{k}_2 &= h\mathbf{f}(t_k + \tfrac{1}{2}h, \mathbf{x}(t_k) + \tfrac{1}{2}\mathbf{k}_1) \\ \mathbf{k}_3 &= h\mathbf{f}(t_k + \tfrac{1}{2}h, \mathbf{x}(t_k) + \tfrac{1}{2}\mathbf{k}_2) \\ \mathbf{k}_4 &= h\mathbf{f}(t_k + h, \mathbf{x}(t_k) + \mathbf{k}_3)\end{aligned} \tag{9}$$

Naloge

1. Najprej izvedite simulacijo sistema z uporabo Eulerjeve integracijske metode (enačba 5). Celotna rešitev naj bo v eni skriptni datoteki, integracijski korak naj bo 10 s, čas simulacije pa 2000 s.

2. Program popravite tako, da bo zgrajen modularno. Poleg glavnega programa boste namreč realizirali funkcije `deriv`, `Euler` in `output`. Funkcija `deriv` naj izračuna odvode stanj. Prva vrstica funkcija naj bo:

   ```
   function dx=deriv(t,x,par)
   ```

   Prvi argument funkcije je trenutna vrednost časa, drugi argument je trenutna vrednost stanj (obeh višin vode), tretji argument je vrednost procesnih parametrov, funkcija pa vrača odvod vektorja stanj. V našem primeru bo `par` vektor štirih procesnih parametrov C_{H1}, C_{H2}, k_{V1} in k_{V2}. Funkcija `Euler` je poseben primer generične funkcije `integ`, ki izvede integracijo po Eulerjevi integracijski metodi. Prva vrstica funkcije naj bo:

   ```
   function [cas,xx]=Euler(x0,par,krm)
   ```

   Prvi argument funkcije je začetno stanje, drugi argument je vektor procesnih parametrov, ki je enak kot zgoraj, tretji argument pa je vektor krmilnih parametrov simulacije, ki obsega dva elementa, in sicer dolžino simulacijskega teka (2000 s) in integracijski korak (10 s). Funkcija vrne časovni vektor in matriko, ki ima toliko vrstic kot časovni vektor. V vsaki vrstici sta dva elementa, in sicer višini vode v obeh shranjevalnikih ob določenem časovnem trenutku. V funkciji `Euler` je for-zanka, ki teče po času. Znotraj vsakokratnega prehoda

zanke se najprej izvede en klic funkcije `deriv` in en klic funkcije `output` (ta funkcija je prisotna le za bolj realističen izgled programa; funkcija ne vrača ničesar, v njej se na zaslon izpisuje trenutna vrednost časa in obeh višin vode – v pravih simulatorjih je funkcija uporabljena za shranjevanje in prikaz podatkov), nato se shranita obe vrednosti višin vode, na koncu pa se izvede enačba (5).

3. Na podoben način kot v prejšnji točki izvedite simulacijo še z Heunovo integracijsko metodo. Funkciji `deriv` in `output` ostaneta nespremenjeni, funkcijo `Heun` pa izvedemo s rahlimi popravki funkcije `Euler`. V zanki sedaj dvakrat kličemo funkcijo `deriv`, kot nakazujeta enačbi (6) in (7).

4. Simulacijo izvedite še z integracijsko metodo Runge-Kutta. Funkciji `deriv` in `output` spet ostaneta nespremenjeni, funkcijo `RK4` pa izvedemo s rahlimi popravki funkcije `Heun`. V zanki sedaj štirikrat kličemo funkcijo `deriv`, kot nakazujeta enačbi (8) in (9).

5. Pri tej nalogi boste analizirali velikost pogreška pri simulaciji. Privzeli bomo, da dobimo pravilni rezultat simulacije, če v Simulinku uporabimo privzete vrednosti simulacijskih parametrov (to shemo ste že realizirali pri prejšnji laboratorijski vaji – naloga 6 in slika 3). Ker boste primerjali simulirano višino vode po tej metodi in eni od integracijskih metod, ki ste jih izvedli pri tej laboratorijski vaji, morate v bloku `To Workspace` nastaviti parameter `Sample time` na takšno vrednost kot pri vsakokratnem klicu »vaše« metode (uporabite isto spremenljivko). Čas trajanja simulacije naj bo 4000 s. V glavnem programu napišite zanko, kjer se spreminja integracijski korak od 10 s do 200 s v korakih po 10 s. V vsakem prehodu skozi zanko izračunajte največjo vrednost absolutne vrednosti razlike med rezultatom Eulerjeve integracijske metode in »pravimi« rezultati, pri čemer analizirajte le rezultate za višino vode v drugem shranjevalniku $h_2(t)$. Izrišite graf, ki na abscisni osi prikazuje integracijski korak, na ordinatni pa vrednost pogreška.

6. Na enak način kot pri prejšnji točki analizirajte še odvisnost napake zaradi končne dolžine integracijskega koraka pri Heunovi metodi.

7. Na enak način kot pri prejšnjih dveh točkah analizirajte še odvisnost napake zaradi končne dolžine integracijskega koraka pri metodi Runge-Kutta.