Ülesanne 4

Ülesande eesmärgiks on aku disainimisel pinge leidmine. Lahendamise käigus kasutatakse MatLabi programmeerimise akend ning tulemuste saamiseks tuleb vajutada RUN nuppu. Vali huvipakkuva aku, koosta graafik ning võrdle tulemusi. Aku pinge muutumist tuleb katsetada erinevatel temperatuuridel näiteks -20˚C, 0˚C ja 20˚C.

Õpiväljundid

Üliõpilane:

- saab teada, millised akuelemendi omadused on vajalikud aku disainimisel;
- saab teada, kuidas muutub aku pinge ja aku laetus erinevatel temperatuuridel;
- õpib Cantera programmi kasutamist/koodi kirjutamist;
- õpib simulatsiooni tulemust analüüsimist ja tõlgendamist.

Ülesande lahendus

Esialgne koodi näidis on leitav Cantera poolt valmistatud MatLabi kasutaja näidistes, kood oli täiendatud ja kohandatud vastavalt püstitud ülesandele. Näidiseks on valitud väiksema drooni aku.

```
SOC = 0:0.5:1; % [-] laetuse oleku vahemik (0%, 50% ja 100%)
I_app = -3; % [A] Väliselt rakendatud vool, drooni aku
% (peab olema negatiivne)
T = 293; % [K] Temperatuur -20C
P = oneatm; % [Pa] Rõhk
```

Joonis 1. Aku disainimisel vajalikud lähteandmed.

Koodi kirjutamisel iga rea lõpus peab seisma semikoolon, muidu programm ei lähe tööle, seda tuleb kindlasti jälgida. Kommentaari saab lisada kasutades protsendi (%) tähist.

```
inputFile = 'lithium_ion_battery.yaml'; % Kasutatav Cantera sisendfail
% (yaml formaadis)
R_elyt = 0.0384; % [Ohm] Elektrolüüdi takistus
S_ca = 1.1167; % [m^2] Katoodi kogu aktiivse materjali pindala
S_an = 0.7824; % [m^2] Anoodi kogu aktiivse materjali pindala

X_Li_an_0 = 0.01; % [-] anood Li moolifraktsioon, kui  SOC = 0 %
X_Li_an_1 = 0.75; % [-] anood Li moolifraktsioon, kui SOC = 100 %
X_Li_ca_0 = 0.99; % [-] katood Li moolifraktsioon, kui SOC = 0 %
X_Li_ca_1 = 0.49; % [-] katood Li moolifraktsioon, kui SOC = 100 %
```

Joonis 2. Aku elemendi omadused.

Joonisel 27 koostatud koodil saab näha, et sisendfailina kasutatakse uue faili, antud kogumiku saab leida Cantera andmehoidlas (joonis 8). Elektrolüüdi takistus, katoodi ja anoodi kogu aktiivse materjali pindala on väga spetsiifilised andmed ning sõltuvad, mis aku tüübi üliõpilane valib.

```
% SOC moolifraktsioonide arvutamise valem
X_Li_an = (X_Li_an_1-X_Li_an_0)*SOC+X_Li_an_0; % anoodi tasakaal
X_Li_ca = (X_Li_ca_0-X_Li_ca_1)*(1-SOC)+X_Li_ca_1; % katoodi tasakaal

% Cantera faaside importimine
anode = Solution(inputFile, 'anode');
cathode = Solution(inputFile, 'cathode');
elde = Solution(inputFile, 'electron');
elyt = Solution(inputFile, 'electrolyte');
anode_interface = Interface(inputFile, 'edge_anode_electrolyte', anode, elde, elyt);
cathode_interface = Interface(inputFile, 'edge_cathode_electrolyte', cathode, elde, elyt);

% Kõikide faaside kindel temperatuur ja rõhk
set(anode,'T',T,'P',P);
set(cathode,'T',T,'P',P);
set(elde,'T',T,'P',P);
set(elyt,'T',T,'P',P);
set(anode_interface,'T',T,'P',P);
set(cathode_interface,'T',T,'P',P);
```

Joonis 3. Arvutuskäigu esimene pool.

Joonisel 28 koostatud koodilt saab näha, et esialgselt tuleb kasutada moolifraktsioonide arvutamise valemit, et tasakaalustada anoodi ja katoodi. Tähis (*) tähendab korrutamist. *Solution* meetodi abil saab teha kindlaks, mida täpsemalt hakatakse kasutama püstitatud ülesanne lahendamiseks, selle jaoks kasutatakse ennem valitud sisendfaili (*inputFile*). Meetod *Interface* määrab sisendfaili, antud sisendfaili kindla kasutatava mudeli ning ülejäänud kõrvalfaase (*anode/cathode, elde, elyt*). Järgmises lõigus määratakse iga faasi kindla temperatuuri ja rõhku, antud ülesanne puhul temperatuur on 253K ja rõhk on 1atm.

```
V_cell = zeros(length(SOC),1);
phi_l_an = 0;
phi_s_ca = 0;
for i = 1:length(SOC)% State of Charge
    % Anoodi elektroodi potentsiaal võrdub 0-ga
    phi_s_an = 0;

    % Anoodi elektrolüüdi potentsiaali arvutamise valem
    phi_l_an = fzero(@(E) anode_curr(phi_s_an,E,X_Li_an(i),anode,elde,elyt,anode_interface,S_an) ...
        -I_app, phi_l_an);

    % Katoodi elektrolüüdi potentsiaali arvutamise valem
    phi_l_ca = phi_l_an + I_app*R_elyt;

    % Katoodi elektroodi potentsiaali arvutamise valem
    phi_s_ca = fzero(@(E) cathode_curr(E,phi_l_ca,X_Li_ca(i),cathode,elde,elyt,cathode_interface,S_ca) ...
        -I_app, phi_s_ca);

    % Elemendi pinge arvutamise valem
    V_cell(i) = phi_s_ca - phi_s_an;
end
```

Joonis 4. Arvutuskäigu teine pool.

Joonisel 29 toodud koodil on näha arvutuskäigu teine osa, kus arvutamine käib iga elemendi pinge sisendvektori sisestuse jaoks eraldi. Tähis „l“ tähendab elektrolüüdi ning tähis „s“ elektroodi. Antud arvutamiskäigus kasutatakse funktsiooni „for“, selle funktsiooni abil saab korrata avaldusi teatud arv kordi. Üldiselt, peale iga funktsiooni (MatLab värvib neid sinisega) peab olema „end“. Ilma koodilõpu tähiseta ei lähe kood tööle. Juhul, kui koodi kirjutamisel tekkis mingi viga, siis MatLab näitab käsuaknas kindla rea, mida tuleb parandada.

```
% Graafik, laetuse oleku funktsiooni põhjal
figure(1);
plot(SOC*100,V_cell,'linewidth',2.5)
ylim([2,5])
xlabel('Laetuse olek / %')
ylabel('Akuelemendi pinge / V')
set(gca,'fontsize',14)
```

Joonis 5. Graafiku loomine.

Joonisel 30 on toodud koodi näidis, kuidas eelmiste andmete ja kalkulatsioonide põhjal saab teha graafiku. Eelmisel joonisel oli välja toodud aku elemendi pinge arvutamise valem, saadud tulemust saab nüüd kasutada graafiku loomisel. Funktsiooni *plot* abil määratakse, mida peab graafik sisaldama (laetuse olek kuni 100%, välja arvutatud pinge ja liini paksus).

Joonistel 31 ja 32 on toodud abifunktsioonid, mille abil tagastatakse anoodi -ja katoodivoolu. Abifunktsioonide sisestamine on kohustuslik, ilma nende funktsioonideta kood ei tööta.

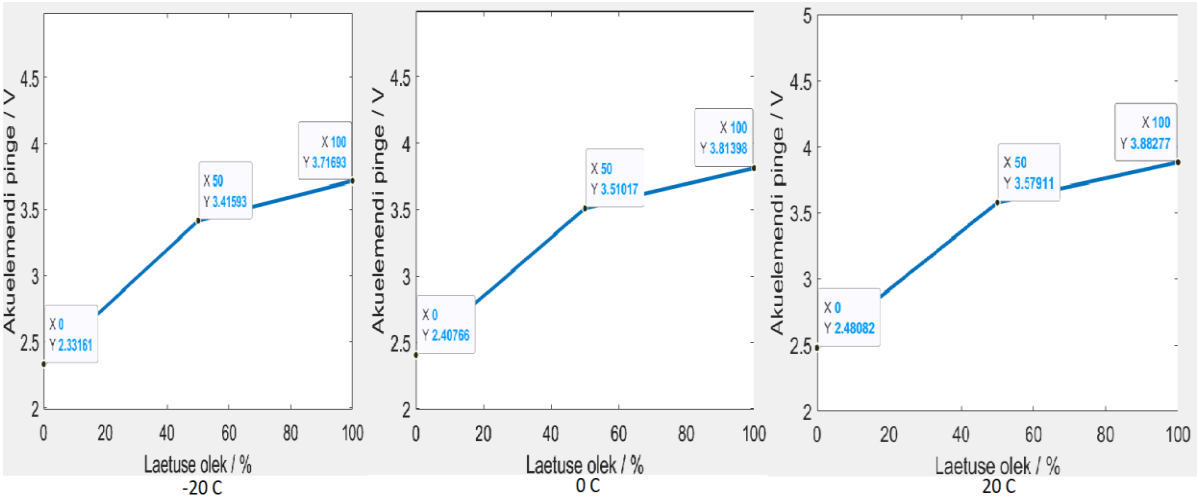
```
function anCurr = anode_curr(phi_s,phi_l,X_Li_an,anode,elde,elyt,anode_interface,S_an)
    % Aktiivse materjali moolifraktsiooni määramine
    set(anode,'X',['Li[anode]:' num2str(X_Li_an) ', V[anode]:' num2str(1-X_Li_an)]);

    % Elektroodi ja elektrolüüdi potentsiaali määramine
    setElectricPotential(elde,phi_s);
    setElectricPotential(elyt,phi_l);

    % Netoreaktsiooni kiirus (anoodipoolne)
    % Reaktsioon vastavalt cti sisendfailile: Li+[elyt] + V[anode] + electron <=> Li[anode]
    r = rop_net(anode_interface); % [kmol/m2/s]

    % Voolu arvutamine (peab olema tühjenemise ajal negatiivne)
    anCurr = r*faradayconstant*S_an; %
end
```

Joonis 6. Abifunktsioonid 1.

```
function caCurr = cathode_curr(phi_s,phi_l,X_Li_ca,cathode,elde,elyt,cathode_interface,S_ca)
    % Aktiivse materjali moolifraktsiooni määramine
    set(cathode,'X',['Li[cathode]:' num2str(X_Li_ca) ', V[cathode]:' num2str(1-X_Li_ca)]);

    % Elektroodi ja elektrolüüdi potentsiaali määramine
    setElectricPotential(elde,phi_s);
    setElectricPotential(elyt,phi_l);

    % Netoreaktsiooni kiirus (katoodipoolne)
    % Reaktsioon vastavalt cti sisendfailile: Li+[elyt] + V[cathode] + electron <=> Li[cathode]
    r = rop_net(cathode_interface); % [kmol/m2/s]

    % Voolu arvutamine (peab olema tühjenemise ajal negatiivne)
    caCurr = r*faradayconstant*S_ca*(-1); %
end
```

Joonis 7. Abifunktsioonid 2.

Tulemuste analüüs



Joonis 8. Saadud tulemused, pinge muutus temperatuuridel -20˚C, 0˚C ja 20˚C.

Joonisel 33 on näha, et aku maksimaalset pinget saab näha, siis kui aku laetuse olek on 100% ning minimaalse pinge näit esineb aku tühjenemisel. Temperatuuri muutus samuti põhjustab pinge vähenemist või kasvamist, näiteks temperatuuril -20C, laetuse olekul 50% pinge näit on 3.41V, kuid temperatuuril +20C, sama laetuse olekul on see näit juba 3.58V. Tudeng ei pea kasutama sama aku andmeid, vaid saab disainida/simuleerida enda valitud akut ning graafikute abil võrrelda tulemusi.