Luku 5

API/MEX ja GUI

Tässä luvussa käsitellään lyhyesti kahta MATLABiin liittyvää toimintakokonaisuutta: *i*) kuinka MATLAB voi toimia muiden ohjelmien kanssa yhdessä, *ii*) miten MATLABilla voidaan rakentaa graafisia käyttöliittymiä. Vaikka asiat käydäänkin läpi lähinnä "manuaalinomaisesti", on tässäkin yhteydessä tärkeää panna merkille vaaditut toimintatavat ja niitä tukevat työkalut, jotka MATLAB-ympäristö näihin tarkoituksiin tarjoaa.

5.1 Application Program Interface (API)

Hyvä ohjelmointiympäristö sisältää kattavan joukon valmista tavaraa, mutta uusia ohjelmia kehitettäessä tulee ennemmin tai myöhemmin vastaan tilanne, jossa olemassaolevat rutiinit eivät sellaisenaan riitä halutun toiminnan aikaansaamiseksi. Tällöin tarvitaan mahdollisuus käytetyn ympäristön täydentämiseen omilla "komponentteilla". Ydinjärjestelmän (meillä MATLAB) ja tällaisten ulkoisten lisärutiinien väliseen kommunikointiin tarvitaan jonkinlainen sopiva rajapinta, joka MATLABissa on nimeltään *Application Program Interface*, lyhyesti API. Yleisimpiä tapauksia, jolloin MATLABin täydentäminen ja toimintojen "ulkoistaminen" tulee ajankohtaiseksi, ovat:

- tarve käyttää jotain valmista aliohjelma(kirjasto)a MATLABista käsin ,
- tarve nopeuttaa MATLABin toimintaa aikakriittisissä sovelluksissa, yleensä *for*-silmukoiden yhteydessä (MATLAB on tulkkaava ympäristö, joten silmurakenteet pyörivät normaaleihin ohjelmointikieliin verrattuna huomattavasti hitaammin).

MLP-verkkojen yhteydessä kyseessä on jälkimmäinen tapaus: data-silmukan läpikäynti kustannusfunktion arvon ja tarvittavien derivaattojen laskemiseksi tekee optimoinnista tuskastuttavan hidasta!

MATLABin API tukee seuraavia toimintoja:

- C- ja Fortran-ohjelmien kutsuminen MATLABista käsin,
- muuttujien vienti MATLABista ulos ja tuonti MATLABiin sisään,
- asiakas/palvelin-arkkitehtuurin muodostaminen MATLABin ja muiden ohjelmistojen välille.

Muuttujien välitysmahdollisuuksien johdosta ulkoisista aliohjelmista voidaan kutsua MATLABin omia rutiineja haluttujen toimintojen (manipulaatioiden) suorittamiseksi. Ulkoisia C- ja Fortran-kielisiä ohjelmia voidaankin liittää MATLABiin siten, että ne näyttävät aivan tavallisilta M-funktioilta. MATLABin yhteydessä näitä dynaamisesti ympäristöön linkittyjä aliohjelmia kutsutaan MEX-tiedostoiksi (EXternal M-files).

MATLAB**in objektityyppi: Vektori**

Jotta MEX-tiedostojen "formaattia" voidaan ymmärtää, täytyy meillä olla jonkinlainen kuva siitä, minkälaisia erityyppiset muuttujat MATLABissa oikein ovat. MATLABin kaikki erilaiset tuetut tietotyypit esitetään käyttäen pelkästään yhdentyyppistä objektia: MATLAB vektoria (MATLAB Array). Tämä tarkoittaa sitä, että käytännössä kaikki MATLABin muuttujat (skalaarit, vektorit, matriisit, merkkijonot, soluvektorit (cell arrays) ja tietueet) talletaan MATLABin vektoriluokkaan. Luokkamääritys `mxArray` vastaa MATLABin sisäistä tietorakennetta tällaisen vektorityypin esitykselle, ja se sisältää tiedot mm. muuttujan nimestä, dimensiosta, tyypistä, sekä siitä, onko kyseessä reaalinen vai kompleksinen rakenne. Työtilassa olevien muuttujien rakennetta voidaan tutkia tarkemmin valmiin rutiinin (oikeasti MEX-tiedosto) `explore` avulla (`x = 2; explore(x)`). Esimerkiksi jonkun reaalisen $n \times m$-matriisin $\mathbf{A}$ esitys `mxArray` rakenteessa pitää sisällään listan matriisin komponenteista, jotka on talletettu *sarakkeittain* järjestyksessä $w_{11}, w_{21}, \ldots, w_{n1}, w_{12}, \ldots, w_{1n}, \ldots, w_{nm}$ (tämä on Fortranin, jolla MATLAB on alunperin implementoitu, talletusjärjestys). Lisäksi `mxArray` pitää sisällään matriisien rivien n ja sarakkeiden m lukumäärät. Huomaa, että tämän talletusmuodon johdosta esim. `reshape`-operaatio ei koske mitenkään varsinaisiin talletettuihin alkioihin (matriisin komponentteihin), vaan se pelkästään muuttaa dimension määrittävien kokonaislukujen arvoja `mxArray`-talletusrakenteessa.

API-rajapinta tarjoaa käytännössä joukon saantimetodeja (access/get methods), joiden avulla `mxArray` rakenteita voidaan lukea ja muokata ulkoisista ohjelmista käsin. Tämän vuoksi tällaisten metodien (rutiinien) nimet alkavat aina joko etuliitteellä (prefix) `mx` tai `mex`. Kun ulkoinen ohjelma haluaa lukea tai muokata jotain MATLABin sisäistä muuttujaa, alkaa metodin nimi etuliitteellä `mx` , esimerkiksi `mxGetPr(plhs[0])`. Kun tietoa puolestaan siirretään ulkoisesta ohjelmasta MATLABin työtilaan, alkaa vastaavan metodin nimi etuliitteellä `mex`, esimerkiksi `mexErrMsgTxt(‘‘Error ...’’)`.

Esimerkkejä

Tarkastellaan seuraavaksi esimerkkien avulla MEX-hommien muotoa ja toimintaa käytännössä. Joitakin valmiita esimerkkejä ulkoisista tiedostoista löytyy alihakemistosta `extern/examples/mex` , jonka hakemistopolun alkuosa pitäisi tarvittaessa löytyä ympäristömuuttujasta <matlab> (Windows) tai $MATLAB (Unix). Jokaisen MEX-tiedoston rakenne koostuu kahdesta eri osasta:

- "gateway"-rutiineista, joiden avulla tarvittavat muuttujat välitetään MATLABista ulkoiseen aliohjelmaan ja takaisin,
- toimintaosasta, jossa MEX-tiedostossa suoritettavat varsinaiset operaatiot on implementoitu.

Sigmoidi-tyyppistä aktivaatiofunktiota soveltava M-funktio `sigmoidi.m` *kiinnitetylle k*:n arvolle (yksinkertaisuuden vuoksi) on muotoa:

```
function [b,c] = sigmoidi1(a,k)
b = 1./(1 + exp(-k*a)); c = k*b.*(1-b);
```

Alla on annettu vastaavan operaation toteutus ulkoisen C-ohjelman `sigmoidi_c.c` avulla. Muuttujien nimeäminen perustuu siihen, että M-funktion formaatissa input-muuttujat `a` ja `k` ovat yhtäsuuruusmerkin oikealla puolella (*right-hand-side rhs*) ja output-muuttujat `b` ja `c` vasemmalla puolella (*left-hand-side lhs*).

```
#include "mex.h"
#include <math.h>

 void sigmoidi (double *a, double *b, double *c, double k, int n)
 {
  int i;
  double t;

  for (i=0; i<n; i++) {
    t = k * *(a + i);
    t = 1/(1 + exp(-t));
    *(b+i) = t;
    *(c+i) = k * t * (1 - t);
  }
  return;
 }

/* The gateway function. */
 void mexFunction(
   int nlhs, mxArray *plhs[],
   int nrhs, const mxArray *prhs[])
   {
   int n, m;
   double k, *a, *b, *c;

/* Check for proper number of arguments */
   if (nrhs !=2) {
     mexErrMsgTxt("Two inputs required.");}
   else if (nlhs !=2) {
     mexErrMsgTxt("Only two outputs is allowed.");
  }

/* Create a pointer to a copy of the first input, matrix a.
   Test to ensure that a is a vector omitted here. */
   a = mxGetPr(prhs[0]);

/* Get the dimensions of the matrix input a. */
   n = mxGetN(prhs[0]);
   m = mxGetM(prhs[0]);

/* Create a scalar copy of the second input, integer k. */
```

```
   k = mxGetScalar(prhs[1]);

/* Create the outputs b and c. */
   plhs[0] = mxCreateDoubleMatrix(n,m,mxREAL);
   b = mxGetPr(plhs[0]);
   plhs[1] = mxCreateDoubleMatrix(n,m,mxREAL);
   c = mxGetPr(plhs[1]);

/* Call the C subroutine. */
   sigmoidi(a, b, c, k, m*n);

}
```

C-ohjelma `sigmoidi_c.c` käännetään MATLABin komentotilassa komennolla

```
>> mex sigmoidi_c.c
```

jonka seurauksena työhakemistoon syntyy uusi *exe*-tiedosto `sigmoidi_c.mexhp7*` (hp). Tämän jälkeen ulkoista ohjelmaa voidaan kutsua normaalin M-funktion tapaan. Testauksen kannalta tilanne, jossa sama operaatio on toteutettu toisaalta MATLABin M-funktiona ja toisaalta ulkoisena ohjelmana tarjoaa kätevän tavan validoida tehty MEX-koodi allaolevaan tyyliin:

```
k = 3; a = 5*(rand(17,1)-0.5); [b1,c1] = sigmoidi(a,k);
% The c-routine below must be precompiled before usage: mex sigmoidi_c.c
[b2,c2] = sigmoidi_c(a,k); norm(b1-b2), norm(c1-c2)
```

Toisaalta, esimerkiksi koko kustannusfunktion ja gradientin MLP-verkon opetuksessa laskevasta MEX-rutiinista voitaisiin kutsua myös MATLABia seuraavasti:

```
   #include "mex.h"
   #include <math.h>
/* Initialization of variables */
   int num_out, num_in;
   mxArray *output_array[2], *input_array[2];
/* ...
   Preparation of variables and call to MatLab */
   num_out = 2;
   num_in = 2;
   input_array[0] = a;
   input_array[1] = *k;
   mexCallMATLAB(num_out,output_array,num_in,input_array,''sigmoidi'');
/* As a result: b = output_array[0] and c = output_array[1] equivalently to
   [b,c] = sigmoidi(a,k) */
```

Näin olemme huomanneet, että toiminnaltaan M- ja MEX-tiedostot voivat olla (ja usein ovatkin) samanlaisia, mutta niiden välillä on myös muutamia peruseroja, jotka kannattaa muistaa:

- MEX-tiedostossa voi olla staattisia muuttujia, mutta M-tiedostoissa ei,
- MEX-tiedostoista käsin voidaan muokata kutsuvan ohjelman työtilassa olevia muuttujia, mutta M-tiedostoista ei.

Tiedon tuonti/vienti MATLABiin/MATLABista

MAT-tiedostot mahdollistavat MATLABin itsensä käyttämässä formaatin olevien muuttujien (datan) tallettamisen levylle esimerkiksi muita ohjelmia varten. Tällä tavalla tietoa voidaan siirtää MATLABista muihin ohjelmiin tai niistä MATLABiin. MAT-formaatissa olevan datan lukua ja kirjoitusta varten peruskomentoja ovat `load` ja `save`. Myös MAT-formaatin käsittely onnistuu ulkoisista ohjelmista käsin, joissa siten voidaan jokin spesiaalimuodossa oleva data muokata MATLABin ymmärtämään muotoon tarvittaessa. Palautetaan tässä yhteydessä mieliin, että MATLABissahan oli myös muita valmiita rutiineja määrämuotoisen tiedon lukemista ja kirjoittamista varten. Lisäksi löytyvät normaalin C-kielen omaiset perusrutiinit tiedostojen käsittelyä (avaus,sulkeminen,formatoitu luku ja kirjoitus yms.) varten. Tästä syystä määrämuotoisten tiedostojen konvertointi MATLABin ymmärtämään formaattiin kannattaa suorittaa ulkoisissa ohjelmissa vain, jos sopivat rutiinit ovat jo valmiina olemassa.

MATLABin etäkäyttö (MATLAB Engine)

MATLAB *Engine Library* mahdollistaa MATLABin etäkäytön mistä tahansa (C- tai Fortran) ohjelmasta käsin. Tässä yhteydessä asiaa ei tämän enempää käsitellä, sillä kiinnostuneet löytävät tarvittavat tiedot MATLABin API-oppaasta!

5.2 Graphical User Interface (GUI)

Erilaiset graafiset käyttöliittymät ovat nykyisin arkipäivää, sillä valtaosa tietokoneiden hyötykäyttäjistä ei enää osaa/halua käyttää mitään tekstipohjaista järjestelmää. Myös MATLAB tarjoaa yksinkertaiset ja kohtuullisen helppokäyttöiset työkalut graafisten käyttöliittymien suunnittelua ja toteuttamista varten.

Yleisistä periaatteista

Graafisten käyttöliittymien rakennetta, suunnittelua ja toteuttamista käsitellään monissa kirjoissa ja useilla opintojaksoilla. Perusperiaatteena on joka tapauksessa aina syytä pitää mielessä seuraava karakterisointi:

> **Käyttöliittymä** on *kokonaisuus*, jonka avulla *käyttäjä* voi suorittaa haluamansa *tehtävät*.

Siksi käyttöliittymää suunniteltaessa on syytä pitää mielessä ainakin seuraavat peruskysymykset:

1. Tietääkö käyttäjä joka tilanteessa missä hän on?
2. Tietääkö käyttäjä joka tilanteessa mitä tapahtuu seuraavaksi?

Onnistuneen (käyttöliittymä)tuotteen keskeinen ominaisuus on sen käytettävyys. Käytettävyyttä voidaan suunnitella (tekovaihe) ja testata (valmis liittymää) esim. seuraavien käsitteiden pohjalta:

Opittavuus (learnability): Aloittelijan käyttämä aika kohtuullisen käyttötaidon oppimiseen.

Tehokkuus (efficiency): Harjaantuneen käyttäjän käyttönopeus.

Muistettavuus (memorability): Käyttötaidon säilyminen; satunnaisen käyttäjän kyky muistaa aiemmin opittu tuotteen käyttötapa.

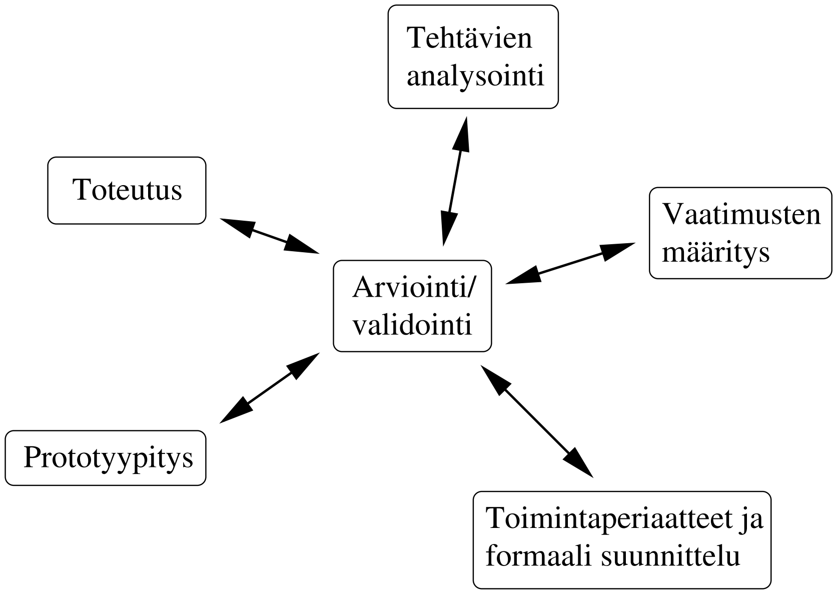
Virheettömyys (faultlessness): Virheiden (joita on joka tapauksessa) jako korjattavissa oleviin, joista voidaan käytön aikana toipua, sekä tuhoisiin, jotka hävittävät tehdyn työn lopullisesti.

Tyytyväisyys (satisfaction): Käyttäjien subjektiiviset arviot tuotteen käyttämisen miellyttävyydestä sekä sen lähestyttävyydestä (approachbility) ennen varsinaisen käytön alkua.

Joustavuus (flexibility): Kuinka liittymä taipuu uusiin tehtäviin ja ympäristöihin?

Vaikuttavuus (effectiveness): Miten hyvin käyttäjä saavuttaa käytölle asettamansa tavoitteet?

Tässä suhteessa ei kannata keksiä pyörää uudestaan, sillä edellämainittujen asioiden testaamiseen on olemassa paljon erilaisia valmiita kyselylomakkeita. Ohjelmankehityksen paradigmana käyttöliittymiä suunniteltaessa käytetään usein ns. *spiraalista tähtimallia*, joka perustuu seuraavien osavaiheiden iterointiin:



Luonnollisesti käytettävyys voi olla (ja sen myös tulisi olla keskeinen) osa liittymän arviointi/validointi-prosessia.

GUIn suunnittelu ja toteutus MATLABilla

MATLABin GUI-työkalujasarjan muodostavat:

- Property Editor (`propedit`): objektien ominaisuuksien modifiointi,
- Guide Control Panel (`ctlpanel`): GUIn rakentamista hallitsevan ja konrolloivan työkalun käyttöliittymä,

- Callback Editor (`cbedit`): toimintojen liittäminen GUIn sisältämiin näkyviin osiin,
- Alignment Tool (`align`): GUIn eri osien sijoittelun viimeistely (tasaus),
- Menu Editor (`menuedit`): Menujen editointi ;-)

Koko kalupakkia pääsee käyttämään komennon `guide` seurauksena.
GUIn rakenteminen MATLABilla perustuu hyvin yksinkertaiseen toimintamalliin:

1. Suunnitellaan liittymän ulkoasu halutunlaiseksi: `propedit & guide & Alignment Tool`,
2. Lisätään käytettyihin elementteihin niiden toiminnot: `Callback and Menu Editors`.

Koko homman lähtökohtana on luonnollisesti MATLABin graafiset objektit, jotka pääosin esiteltiin luvussa 2. Tutustutaan seuraavaksi lähemmin graafisen objektin `Uicontrol` rakenteeseen, sillä tämä määrittää GUIn sisältämien perustoimintojen tyypit. `Figure`-ikkunoiden yläpalkkien sisältämien valikkorakenteiden sisällön ja käsittelyn mahdollistava objekti `Uimenu` rakentuu samaan malliin, joten sen käsittely jätetään tässä yhteydessä oman opiskelun varaan. `Uicontrol`-objektin sisältämien atribuuttien johdosta käyttöliittymä voi sisältää seuraavanlaisia valmiiksi tuettuja toimintoja:

Push buttons: Kuten puhelimen näppäimet. Jokainen tällaisen objektin painallus hiiren avulla aktivoi siihen liitetyn toiminnon (`ButtonDownFcn`-ominaisuus, esim. "OK" painonappi).

Check boxes: Myös jokainen tällaisen objektin painallus hiiren avulla aktivoi siihen liitetyn toiminnon, mutta toisin kuin edellinen objekti, tämä jää painamisen jälkeen osoittamaan kyseistä tilaa. Näiden avulla voidaan ohjelman toimintaa muuttaa kahden eri tilan välillä.

Pop-up menus: Yleensä ikkunan yläpalkin sisältämät toiminnot, jotka painettaessa avautuvat näyttämään listan mahdollisia alivalintoja.

Radio buttons: Yksittäisistä “Check box” rakenteista muodosttu joukko erillisiä toimintoja, joista vain yksi voi olla tiettynä ajanhetkenä aktiivisena.

Sliders: Annetulla välillä sijaitsevan numeerisen arvon asettaminen hiiren avulla “liuvuttamalla”.

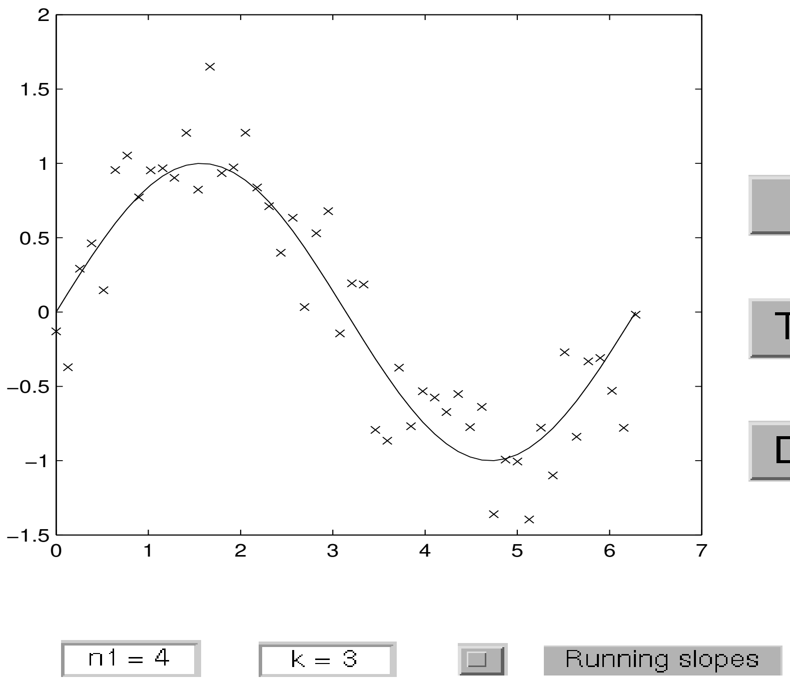
Editable text: Mahdollistaa käyttäjän antaman tekstin luvun.

Static text: Muuttumaton tekstinpätkä, käytetään yleensä otsikoiden yms. asettamista varten.

Frames: Figure-ikkunan halutun osan/osien korostaminen käytön helpottamiseksi. Näihin ei voi liittää mitään varsinaista toiminnallisuutta eli jotain kutsuttavaa rakennetta.

List boxes: Lista merkkijonoja (esim. tiedostojen nimiä), joista voidaan hiirellä valita haluttu.

Tarkastellaan työkalujen kanssa toimimista seuraavan esimerkin avulla, joka tullaan käymään läpi myös harjoituksissa. Kohinaista sini-funktiota approksimoivan MLP-verkon käyttöliittymä voisi olla esimerkiksi seuraavanlainen:



Liittymän toiminnot jakaantuvat seuraavalla tavalla:

Init (`pushbutton`) : Luodaan satunnainen esimerkkidata, piirretään se näkyviin ja suoritetaan esiskaalaus.

Train (`pushbutton`) : Opetetaan MLP-verkko skaalatun datan avulla ratkaisemalla optimointitehtävä.

Draw (`pushbutton`) : Piirretään MLP-verkon oppima funktio näkyviin toisella pisteistöllä.

n1 = 4 (`edit`) : Käyttäjän muutettavissa oleva merkkijono, jonka avulla määritetään piilokerroksen koko.

k = 3 (`edit`) : Käyttäjän muutettavissa oleva merkkijono, jonka avulla määritetään piilokerroksen sigmoidin muoto (slope).

Running slopes (`checkbox`) : Jos tämä on "päällä", käytetään ns. juoksevaa sigmoidin `k`:n valintaa.

Käytännössä GUIn toteuttaminen tapahtuu siis seuraavasti:

1. Toteuta GUIn ulkoasu `guide`n (toimintojen valinta ja sijoittelu) ja `propedit`in (esim. nappien tekstit ja tekstifontit) sekä `Alignment tool`in avulla.

2. Liitä näkyviin objekteihin halutut toiminnot `Callback Editor`in avulla.

3. Aktivoi liittymä `guide`sta (hiirellä) ja talleta se esim. liittymän ikkunan yläpalkista. Tämän seurauksena syntyy kaksi tiedostoa (GUI $\sim$ valittu nimi): GUI.m ja GUI.mat, joista ensimmäinen sisältää liittymän varsinaiset toiminnot ja toinen siihen mahdollisesti

liittyvän datan. On erittäin hyödyllistä katsoa syntynyttä M-tiedostoa, sillä sen rakenne kertoo parhaiten ne tavat, joilla MATLAB liittymän varsinaisesti muodostaa graafisten objektien avulla. Kunhan hommasta pääsee jyvälle, nopein tapa tehdä muutoksia olemassaoleviin liittymiin onkin suora GUI.m-tiedoston editointi.

4. Testaa ja muokkaa tarvittaessa!